Index

PIRENS Institute of Business Management and Administration, Loni BK.		
Roll Number:ALEX CARRY	**Sign:**	**Date: / /**
Student Name: Alex carry		
Subject Name: Python		
Program 1 : Python installation and configuration with windows and Linux.		

Windows Installation

Method 1: Using the Official Installer

1. **Download the Installer:**
   - Visit the official Python website (https://www.python.org/downloads/) and download the latest Python installer for Windows.
2. **Run the Installer:**
   - Double-click the downloaded installer to start the installation process.
3. **Customize Installation:**
   - Choose the desired installation options, such as:
     - **Installation Location:** Select the directory where you want to install Python.
     - **Add Python to PATH:** This allows you to run Python from the command prompt without specifying the full path.
     - **Install for All Users:** This option will install Python for all users on the system.
4. **Complete the Installation:**
   - Click the "Install Now" button to start the installation. Follow the on-screen instructions to complete the process.

Method 1: Using the Package Manager

Ubuntu/Debian:

```
sudo apt-get update
sudo apt-get install python3 python3-pip
```

Fedora/CentOS:

```
sudo dnf install python3 python3-pip
```

Method 2: Using the Official Installer

1. **Download the Installer:**
   - Visit the official Python website (https://www.python.org/downloads/) and download the latest Python source code or binary installer for your Linux distribution.
2. **Extract and Install:**
   - Extract the downloaded archive and follow the installation instructions provided in the README file.

Setting Up a Virtual Environment

To isolate Python projects and their dependencies, it's recommended to use virtual environments.

Windows:

```
python -m venv my_env
my_env\Scripts\activate
```

.

Linux:

```
python3 -m venv my_env
source my_env/bin/activate
```

Checking Python Installation

Open a command prompt or terminal and type:

```
python –version
```

Windows :

Python 3.12.6

Linux :

Python 3.12.6

PIRENS Institute of Business Management and Administration, Loni BK.		
Roll Number:ALEX CARRY	**Sign:**	**Date: Date: / /**
Student Name: Alex carry		
Subject Name: Python		
Program Title: 2 Programs for understanding the data types, control Flow statements, blocks and loops.		

```
x = 10  # Integer
y = 3.14  # Float

# Boolean
is_true = True

# String
name = "Alice"

# List (ordered, mutable)
fruits = ["apple", "banana", "cherry"]

# Tuple (ordered, immutable)
colors = ("red", "green", "blue")

# Set (unordered, unique)
vowels = {"a", "e", "i", "o", "u"}

person = {"name": "Bob", "age": 30, "city": "New York"}
```

Control Flow Statements

1. Conditional Statements (if, elif, else)c

```
age = 25
if age >= 18:
 print("You are an adult.")
elif age >= 13:
 print("You are a teenager.")
else:
 print("You are a child.")
```

2. Loops

a. For Loop

Python

```
for i in range(5):
 print(i)

for fruit in fruits:
 print(fruit)
```

b. While Loop

```
count = 0
while count < 5:
 print(count)
 count += 1
```

Blocks

Python uses indentation to define blocks of code. Indentation is typically 4 spaces.

Python

```
if age >= 18:
 print("You are eligible to vote.")
 print("You can also drive.")
```

Example: A Simple Calculator

Python

```
def calculate():
 num1 = float(input("Enter the first number: "))
 num2 = float(input("Enter the second number: "))
 operator = input("Enter the operator (+, -, *, /): ")

 if operator == "+":
  result = num1 + num2
 elif operator == "-":
  result = num1 - num2
 elif operator == "*":
  result = num1 * num2
 elif operator == "/":
  if num2 == 0:
   print("Error: Division by zero")
 else:
  result = num1 / num2
 else:
  print("Invalid operator")

 if result is not None:
   print("Result:", result)

calculate()
```

Output :

Control Flow Example:

Given the age of 25, the output would be:

Loop Examples:

For Loop:

```
"D:\Python programs\
You are an adult.
0
1
2
3
4
apple
banana
cherry
0
1
2
3
4
```

```
"D:\Python programs\pythonProject5\.
You are eligible to vote.
You can also drive.
Enter the first number: 5
Enter the second number: 6
Enter the operator (+, -, *, /): +
Result: 11.0
```

PIRENS Institute of Business Management and Administration, Loni BK.		
Roll Number:ALEX CARRY	**Sign:**	**Date: Date: / /**
Student Name: Alex carry		
Subject Name: Python		
Program Title 3: Programs for understanding functions, use of built in functions, user defined functions.		

Built-in Functions

```
print(abs(-5))
print(round(3.14159, 2))

 # String Functions
text = "Hello, World!"
print(len(text))
print(text.upper())
print(text.lower())

 # List Functions
my_list = [1, 2, 3, 4, 5]
print(max(my_list))
print(min(my_list))
print(sum(my_list))
```

User-Defined Functions

```
def calculate_area(length, width):
    return length * width

length = float(input("Enter the length: "))
width = float(input("Enter the width: "))

area = calculate_area(length, width)
print("The area of the rectangle is:", round(area, 2))
```

Output :

```
"D:\Python programs\
5
3.14
13
HELLO, WORLD!
hello, world!
5
1
15
```

```
"D:\Python programs\pythonProject5\.
Enter the length: 5
Enter the width: 6
The area of the rectangle is: 30.0

Process finished with exit code 0
```

PIRENS Institute of Business Management and Administration, Loni BK.		
Roll Number:ALEX CARRY	**Sign:**	**Date: Date: / /**
Student Name: Alex carry		
Subject Name: Python		
Program Title 4: Programs to use existing modules, packages and creating modules, packages		

```
from math import sqrt, pi

# Using specific functions
radius = 5
area = pi * radius**2
print(area)


import math as m

# Using functions with aliases
result = m.sqrt(25)
print(result)
```

Creating Modules and Packages

Creating a Module:

```
# my_module.py
def greet(name):
    print("Hello, " + name + "!")

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

Using the Module:

```
import my_module

my_module.greet("Alice")
result = my_module.factorial(5)
print(result)
```

Creating a Package:

```
my_package/
    __init__.py
```

module1.py
module2.py
module1.py:

Python
```
def add(x, y):
    return x + y
```

module2.py:

Python
```
def subtract(x, y):
    return x - y
```

Using the Package:

```
import my_package.module1 as m1
import my_package.module2 as m2

result1 = m1.add(5, 3)
result2 = m2.subtract(10, 4)
print(result1)
print(result2)
```

Output :

Importing Specific Functions:

```
========== RESTART: C:\Users
78.53981633974483
```

Using Functions with Aliases:

```
========== RESTART: C:
5.0
```

Creating and Using a Module:

```
====== RESTART: C:/Users/
Hello, Alice!
120
```

Creating and Using a Package:

```
= RESTART: C:\Users\Sanket\
8
6
```

PIRENS Institute of Business Management and Administration, Loni BK.		
Roll Number:ALEX CARRY	**Sign:**	**Date: Date: / /**
Student Name: Alex carry		
Subject Name: Python		
Program Title 5**: Programs for implementations of all object-oriented concepts like class, method, inheritance, polymorphism etc. (Real life examples must be covered for the implementation of object-oriented concepts).**		

Object /Instance program :

```
class Car:
    def __init__(self, brand, model, year):
        self.brand = brand
        self.model = model
        self.year = year

    def start(self):
        print(f"Starting the {self.brand} {self.model}")

# Creating objects
car1 = Car("Toyota", "Camry", 2023)
car2 = Car("Honda", "Civic", 2022)

# Calling methods
car1.start()
car2.start()
```

Inheritance

Creating a Base Class and a Derived Class

```
class Animal:
    def __init__(self, name):
        self.name = name

    def eat(self):
        print(f"{self.name} is eating.")

class Dog(Animal):
    def bark(self):
        print(f"{self.name} is barking.")

# Creating an object of the derived class
dog1 = Dog("Buddy")
```

```
dog1.eat()
dog1.bark()
```

3. Polymorphism
Method Overriding

```
class Shape:
    def area(self):
        pass

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return 3.14 * self.radius * self.radius

class Rectangle(Shape):
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

# Creating objects
circle = Circle(5)
rectangle = Rectangle(4, 3)

print(circle.area())
print(rectangle.area())
def add(a, b=0, c=0):
    return a + b + c

print(add(2, 3))
print(add(2, 3, 4))
```
Use code with caution.

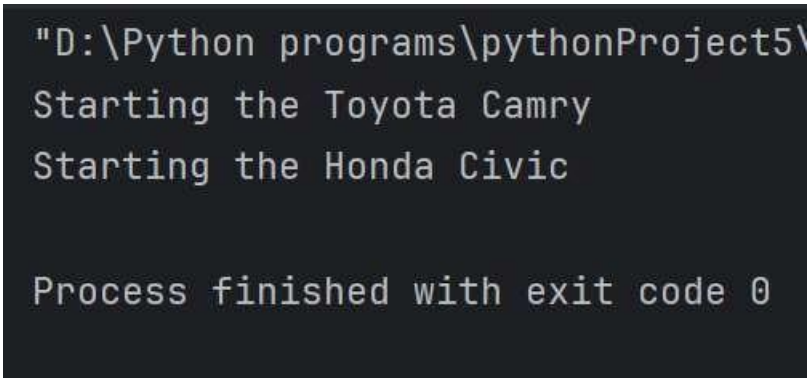
4. Encapsulation

```
class BankAccount:
    def __init__(self, balance):
        self.__balance = balance

    def deposit(self, amount):
        self._balance += amount
        print(f"Deposited {amount}. New balance: {self.__balance}")
```

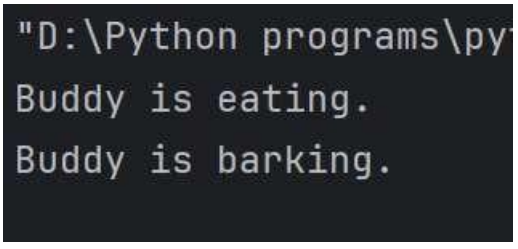
```
    def withdraw(self, amount):
        if self._balance >= amount:
            self._balance -= amount
            print(f"Withdrew {amount}. New balance: {self._balance}")
        else:
            print("Insufficient balance")

account = BankAccount(1000)
account.deposit(500)
account.withdraw(200)
```

OUTPUT :

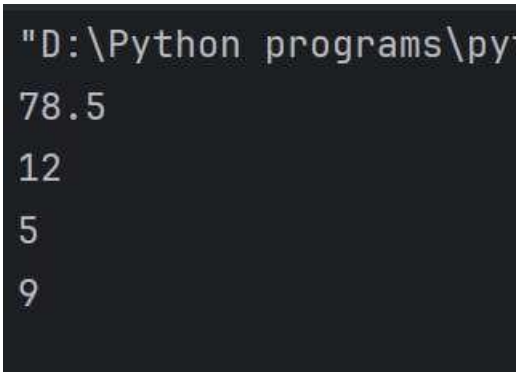
Class and Object:



2. Inheritance:



3. Polymorphism:



4. Encapsulation:

```
"D:\Python programs\pythonProject5\.
Deposited 500. New balance: 1500
Withdrew 200. New balance: 1300

Process finished with exit code 0
```

PIRENS Institute of Business Management and Administration, Loni BK.		
Roll Number:ALEX CARRY	Sign:	Date: Date: / /
Student Name: Alex carry		
Subject Name: Python		
Program Title 6 Programs for parsing of data, validations like Password, email, URL, etc.		

Parsing JSON Data:

Python

```
import json

json_data = '{"name": "Alice", "age": 30, "city": "New York"}'

data = json.loads(json_data)

print(data["name"])
print(data["age"])
print(data["city"])
```

Validations

Password Validation:

```
import re

def validate_password(password):
    if len(password) < 8:
        return False
    if not re.search("[a-z]", password):
        return False
    if not re.search("[A-Z]", password):
        return False
    if not re.search("[0-9]", password):
        return False
    if not re.search("[!@#$%^&*]", password):
        return False
    return True

password = input("Enter your password: ")
if validate_password(password):
```

```
    print("Valid password")
else:
    print("Invalid password")
```

Email Validation:

```
import re

def validate_email(email):
    regex = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
    if re.fullmatch(regex, email):
        return True
    else:
        return False

email = input("Enter your email: ")
if validate_email(email):
    print("Valid email")
else:
    print("Invalid email")
```

URL Validation:

```
import re

def validate_url(url):
    regex = r"^(https?://)?(www\.)?[-a-zA-Z0-9@:%._\+~#=]{1,256}\.[a-zA-Z0-9()]{1,6}\b([-a-zA-Z0-9()@:%_\+.~#?&//=]*)$"
    if re.fullmatch(regex, url):
        return True
    else:
        return False

url = input("Enter a URL: ")
if validate_url(url):
    print("Valid URL")
else:
    print("Invalid URL")
```

OUTPUT :

JSON Parsing:

```
"D:\Python programs\pyth
Alice
30
New York
```

Password Validation:

```
"D:\Python programs\pythonProject5
Enter your password: shree@123
Invalid password
```

Email Validation:

```
"D:\Python programs\pythonProject5\.venv
Enter your email: helloshree@gmail.com
Valid email
```

URL Validation:

```
"D:\Python programs\pythonProject5\
Enter a URL: sanketsahaneofficial
Invalid URL
```

PIRENS Institute of Business Management and Administration, Loni BK.		
Roll Number:ALEX CARRY	Sign:	Date: Date: / /
Student Name: Alex carry		
Subject Name: Python		
Program Title 7 : Programs for Pattern finding should be covered. .		

```
  def is_palindrome(string):
    return string == string[::-1]

string = "racecar"
if is_palindrome(string):
    print(string, "is a palindrome")
else:
    print(string, "is not a palindrome")
```

2 Finding Prime Numbers

```
def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

num = 17
if is_prime(num):
    print(num, "is a prime number")
else:
    print(num, "is not a prime number")
```

3 Finding Anagrams

```
def is_anagram(str1, str2):
    return sorted(str1) == sorted(str2)

str1 = "listen"
str2 = "silent"
if is_anagram(str1, str2):
    print(str1, "and", str2, "are anagrams")
```

```
else:
    print(str1, "and", str2, "are not anagrams")
```

4 Finding Fibonacci Series

Python

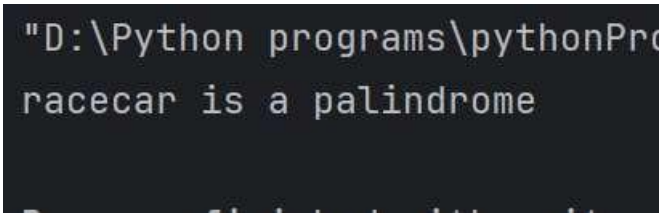
```
def fibonacci(n):
    if n <= 0:
        return []
    elif n == 1:
        return [0]
    else:
        fib_series = [0, 1]
        for i in range(2, n):
            fib_series.append(fib_series[i-1] + fib_series[i-2])
        return fib_series

n = 10
print(fibonacci(n))
```

5 Finding Perfect Numbers

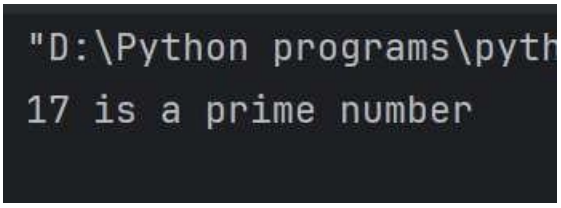
Python

```
def is_perfect_number(num):
    sum_divisors = 0
    for i in range(1, num):
        if num % i == 0:
            sum_divisors += i
    return sum_divisors == num

num = 28
if is_perfect_number(num):
    print(num, "is a perfect number")
else:
    print(num, "is not a perfect number")
```

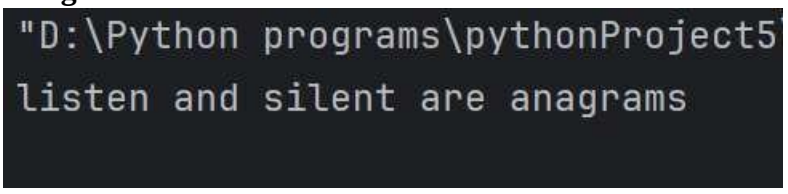
OUTPUT :



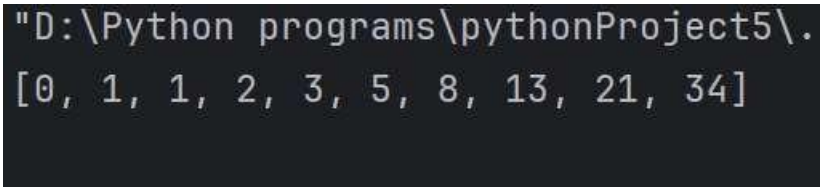
Prime Number :



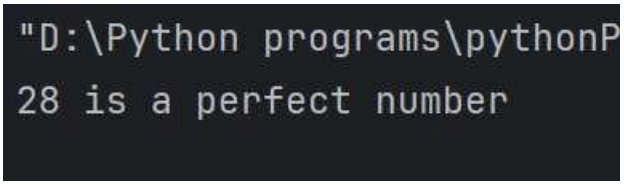
Anagram :



Fibonacci series :



Perfect Number :



PIRENS Institute of Business Management and Administration, Loni BK.		
Roll Number:ALEX CARRY	**Sign:**	**Date: Date: / /**
Student Name: Alex carry		
Subject Name: Python		
Program Title 8: Programs covering all the aspects of Exception handling, user defined exception, Multithreading should be covered		

Exception Handling

```
def divide(a, b):
    try:
        result = a / b
        print("Result:", result)
    except ZeroDivisionError:
        print("Error: Division by zero")
    except TypeError:
        print("Error: Invalid input types")
    else:
        print("Division successful")
    finally:
        print("This block always executes")

divide(10, 2)  # Output: Result: 5.0, Division successful, This block always executes
divide(10, 0)  # Output: Error: Division by zero, This block always executes
divide("a", "b")  # Output: Error: Invalid input types, This block always executes
```

User-Defined Exception

```
class NegativeNumberError(Exception):
    pass

def factorial(n):
    if n < 0:
        raise NegativeNumberError("Factorial is not defined for negative numbers")
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)

try:
    result = factorial(-5)
except NegativeNumberError as e:
    print(e)  # Output: Factorial is not defined for negative numbers
```

Multithreading

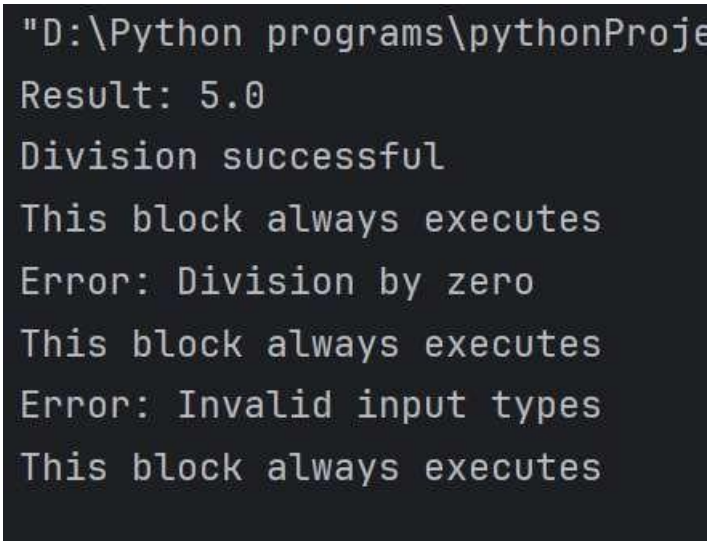
Python.py

```
import threading

def print_numbers():
    for i in range(1, 6):
        print(i)
```

```
def print_letters():
    for char in 'abcde':
        print(char)

t1 = threading.Thread(target=print_numbers)
t2 = threading.Thread(target=print_letters)

t1.start()
t2.start()

t1.join()
t2.join()

print("Both threads have finished")
```

OUTPUT :

Exception Handling:



User-Defined Exception:

```
"D:\Python programs\pythonProject5\.venv\Scripts\
Factorial is not defined for negative numbers
```

Multithreading:

```
"D:\Python programs\pythonProje
1
2
3
4
5
a
b
c
d
e
Both threads have finished
```

PIRENS Institute of Business Management and Administration, Loni BK.		
Roll Number:ALEX CARRY	**Sign:**	**Date: Date: / /**
Student Name: Alex carry		
Subject Name: Python		
Program Title 9 : Basic programs with NumPy as Array, Searching and Sorting, date & time and String handling .		

NumPy

Array Creation:

Python

```
import numpy as np

# 1D array
arr1 = np.array([1, 2, 3, 4, 5])
print(arr1)

# 2D array
arr2 = np.array([[1, 2, 3], [4, 5, 6]])
print(arr2)
```

Array Operations:

```
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])

# Addition
print(arr1 + arr2)

# Subtraction
print(arr1 - arr2)

# Multiplication
print(arr1 * arr2)

# Division
print(arr1 / arr2)
```

Searching and Sorting:

```
arr = np.array([3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5])

# Sorting
sorted_arr = np.sort(arr)
print(sorted_arr)

# Finding unique elements
unique_elements = np.unique(arr)
print(unique_elements)

# Finding indices of specific values
```

```
indices = np.where(arr == 5)
print(indices)
```

Date and Time:

```
import datetime

# Get current date and time
now = datetime.datetime.now()
print(now)

# Specific date and time
specific_date = datetime.datetime(2023, 11, 25, 12, 30)
print(specific_date)

# Formatting dates and times
formatted_date = now.strftime("%d/%m/%Y %H:%M:%S")
print(formatted_date)
```

String Handling:

```
text = "Hello, World!"

# Length of the string
print(len(text))

# Slicing
print(text[0:5])  # Output: Hello

# Concatenation
new_text = text + " How are you?"
print(new_text)

# Finding substrings
index = text.find("World")
print(index)
```

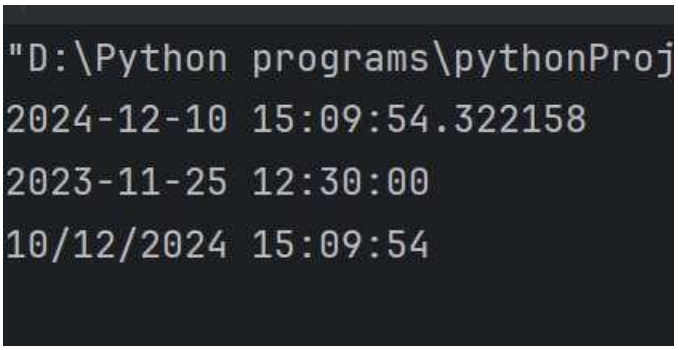
OUTPUT :

NumPy:

```
== RESTART: C:/Us
[1 2 3 4 5]
[[1 2 3]
 [4 5 6]]
```

```
== RESTART: C:/User
[5 7 9]
[-3 -3 -3]
[ 4 10 18]
[0.25 0.4  0.5 ]
```

Date and Time :



String Handling: