THEORY QUESTIONS ASSIGNMENT

Software Stream

Maximum score: 100

Maximum score: 100

KEY NOTES

- This assignment to be completed at student's own pace and submitted before given deadline.
- There are 10 questions in total and each question is marked on a scale 1 Ato 10. The maximum possible grade for this assignment is 100 points.
- Students are welcome to use any online or written resources to answer these questions.
- The answers need to be explained clearly and illustrated with relevant examples where necessary. Your examples can include code snippets, diagrams or any other evidence-based representation of your answer.

Theory questions	**10 point each**

1. How does Object Oriented Programming differ from Process Oriented Programming?

Object Oriented Programming: OOP is object-oriented, the program is divided into objects, Bottom-Up approach Data in each object is controlled on its own. Object functions are linked through message passing. Adding new data and functions is easy.Inheritance is supported in three modes: public, private & protected.

Access control is done with access modifiers. Inheritance is supported in three modes: public, private & protected. Data can be hidden using Encapsulation. Overloading functions, constructors, and operators are done. Classes or

functions can be linked using the keyword “friend, only in C++. The virtual function appears during inheritance. The existing code can be reused. Used for solving big problems. C++, JAVA, VB.NET, C#.NET.

Process Oriented Programming: POP is structure or procedure-oriented. The program is divided into functions. Top-down approach. Every function has different data, so there’s no control over it. Parts of a program are linked through parameter passing. Expanding data and function is not easy. Inheritance is not supported. No access modifiers supported.No data hiding. Data is accessible globally. Overloading is not possible. No friend function. No virtual classes functions. No code reusability. Not suitable for solving big problems.
C, VB, FORTRAN, Pascal

2. What's polymorphism in OOP?

Same name different form. For example, a method can have the same name and can take different parameters. area(radius), area(length,breath). Polymorphism in OOP is a feature in which a method can have the same name but can take different number of parameters and return the result as per the situation. For example,

```
class Language:
  def greet(self):
    raise NotImplementedError("Please implement greet method")

class Nepali(Language):
  def greet(self):
    print("Namaskar")

class Spanish(Language):
  def greet(self):
    print("Hola")
```

```
def intro(lang):
  lang.greet()

nepali= Nepali()
spanish = Spanish()

intro(nepali)
intro(spanish)
```

In the above code, intro method is accepting two different types of parameters

(instances of Nepali and Spanish class) and works fine.

3. What's inheritance in OOP?
Inheritance inherits the properties and functionality from the parent class to the child class.

```
class Language:
  def __init__(self):
    self.country = ""

  def origin(self):
    return self.country

class Nepali(Language):
  def __init__(self):
    self.country = "Nepal"

class Spanish(Language):
  def __init__(self):
    self.country = "Spain"


nepali= Nepali()
spanish = Spanish()

print(nepali.origin())
print(spanish.origin())
```

In the above example, we have defined the origin method in our parent class. Without defining

it in the child classes, we are able to return the desired value by using the inheritance property.

4. If you had to make a program that could vote for the top three funniest people in the office, how would you do that? How would you make it possible to vote on those people?

```
class Voting:
  def __init__(self, nominees_list):
      self.nominees = dict.fromkeys(nominees_list, 0)

  def vote(self, name):
      if list(self.nominees.keys()).count(name) == 0:
          print(f"No nominees found with name {name}")
      else:
          self.nominees[name] += 1

  def results(self):
      sorted_data = sorted(self.nominees.items(), key=lambda x: x[1], reverse=True)
      sorted_data_dict = dict(sorted_data)
      top_three = list(sorted_data_dict.items())[:3]
      for nominee in top_three:
          print(f"{nominee[0]} got {nominee[1]} votes.")


p1 = Voting(["Shiwani", "Emily", "Ali", "Shepstone"])
p1.vote("Shiwani")

p1.results()
```

5. What's the software development cycle?

Planning: The first step in SDLC is requirement gathering and analysis, this is where the top management starts planning the project. The clients provide required information to the management and analysis begins to determine the project approach deliverables and anticipated final outcomes. Business requirements from the customers are gathered to determine who will use the software and how.

Designing: The second step is software design where software architects use the requirements gathered from the first stage to produce several designs of the product. Each design is reviewed in a design document by various stake holders, the best design is then selected.

Building: The third stage is coding and implementation, which is when the documents from the second phase are used to implement the design and produce the code.

Testing: After the building part is done , the testing step can begin. The code is tested based on the customer's requirements to ensure the code works according to specifications.

Deployment: After the fourth stage, deployment occurs. This is when the product is delivered to the customer.

6. What's the difference between agile and waterfall?

- Waterfall is a Liner Sequential Life Cycle Model whereas Agile is a continuous iteration of development and testing in the software development process.
- The Agile methodology is known for its flexibility whereas Waterfall is a structured software development methodology.
- Comparing the Waterfall methodology vs Agile which follows an incremental approach whereas the Waterfall is a sequential design process.
- Agile performs testing concurrently with software development whereas in Waterfall methodology testing comes after the "Build" phase.
- Agile allows changes in project development requirement whereas Waterfall has no scope of changing the requirements once the project development starts.
- Waterfall is a Liner Sequential Life Cycle Model whereas Agile is a continuous iteration of development and testing in the software development process.
- In Agile vs Waterfall difference, the Agile methodology is known for its flexibility whereas Waterfall is a structured software development methodology.
- Comparing the Waterfall methodology vs Agile which follows an incremental approach whereas the Waterfall is a sequential design process.
- Agile performs testing concurrently with software development whereas in Waterfall methodology testing comes after the "Build" phase.
- Agile allows changes in project development requirement whereas Waterfall has no scope of changing the requirements once the project development starts.

7. What is a reduced function used for?

Python offers a function called reduce() that allows you to reduce a list in a more concise way.

Here is the syntax of the reduce() function:

```
reduce(fn,list)
```

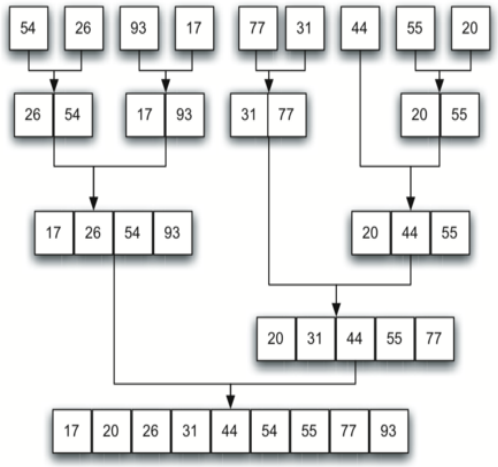
To use the reduce() function, you need to import it from the functools module using the following statement at the top of the file:

```
from functools import reduce
```

8. How does merge sort work

Merge sort separates a list of data into two halves and then, calls these subparts to divide it further into two halves. It repeats the operation until each list component has just one element. By sorting these subparts of one element into two components, it will later merge them together. After sorting, the two-element subpart will be linked with the other two components.

Merge sort is a popular divide-and-conquer sorting method. Merge sort is widely used because of its speed in sorting data. It's one of the better illustrations of how to divide and conquer algorithms that may be used in practice.



9. Generators - Generator functions allow you to declare a function that behaves like an iterator, i.e. it can be used in a for loop. What is the use case?

It is used to make iterators and recursive functions where the code can be really concise and clear.

```
def fibonacci(n):
  a = 0
  b = 1
  for i in range(n):
      yield a
```

```
        a, b = b, a + b

for x in fibonacci(11):
    print(x)
```

10. Decorators - A page for useful (or potentially abusive?) decorator ideas. What is the return type of the decorator?

Decorators is a function that takes function as an argument and modifies the behaviour of that function without modifying the actual function.

```
def my_decorator(func):
    def wrapper():
        print("This is before calling the function")
        func()
        print("This is after calling the function.")
    return wrapper

def say_hello():
    print("hello!")

say_hello = my_decorator(say_hello)

say_hello()
```