



MATURE

PDF

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
VADAPALANI CAMPUS
DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

18CSC207J – ADVANCED PROGRAMMING PRACTICES

NETWORK PROGRAMMING PARADIGM

1. _____ programming is a way of connecting two nodes on a network to communicate with each other.
 - i. Logic
 - ii. **Socket**
 - iii. Functional
 - iv. Symbolic

2. Which method used by the server to initiate the connection with the client?
 - i. Listen()
 - ii. Close()
 - iii. Bind()
 - iv. **Accept()**

3. _____ is a socket through which data can be transmitted continuously.
 - i. Datagram Socket
 - ii. **Stream Socket**
 - iii. Raw Socket
 - iv. Binary Socket

4. _____ paradigm deals with client – server communication
 - i. Dependent type paradigm
 - ii. Parallel programming paradigm
 - iii. Network paradigm
 - iv. Concurrent programming paradigm

5. _____ protocol facilitates sending of datagrams in an unreliable manner.

- i. TCP
 - ii. UDP**
 - iii. HTTP
 - iv. FTP
- 6. Which among methods are not server socket?
 - i. connect()**
 - ii. bind()
 - iii. listen()
 - iv. accept()
- 7. In UDP, Which among methods are used to receive messages at endpoint
 - i. sock_object.recv()
 - ii. sock_object.send()
 - iii. sock_object.recvfrom()**
 - iv. sock_object.sendto()
- 8. In TCP, Which among methods are used to send messages from endpoint
 - i. sock_object.recv()
 - ii. sock_object.send()**
 - iii. sock_object.recvfrom()
 - iv. sock_object.sendto()
- 9. The protocol which defines IPv4 is
 - i. AF_UNIX
 - ii. SOCK_STREAM
 - iii. AF_INET**
 - iv. SOCK_DGRAM
- 10. The correct order of methods used in server socket is
 - i. Socket(), Bind(), listen(), accept()**
 - ii. Socket(), listen(), bind(), accept()
 - iii. Socket(), accept(), bind(), listen()
 - iv. Socket(), bind(), accept(), listen()

11. _____ is a type of network socket which provides connection less point for sending and receiving packets.

- i. **Datagram Socket**
- ii. Stream Socket
- iii. Raw Socket
- iv. Binary Socket

12. _____ do not use any transport protocol but data is directly transmitted over IP protocol

- i. Datagram Socket
- ii. Stream Socket
- iii. **Raw Socket**
- iv. Binary Socket

13. The _____ is a physical path over which the message travels.

- i. Protocol
- ii. **Medium**
- iii. Path
- iv. Route

14. A pair (host, port) is used for the _____ address family.

- i. AF_NETLINK
- ii. AF_INET6
- iii. **AF_INET**
- iv. AF_ALG

15. Use _____ to make the socket to visible to the outside world.

- i. socket.listen()
- ii. socket.visible()
- iii. socket.socket()
- iv. **Socket.gethostname()**

16. In which mode socket is created in default.

- i. **Blocking mode**
- ii. Non-Blocking Mode
- iii. Timeout mode
- iv. Accept mode

17. Which method is recommended to be called before calling connect() method?

- i. getdefaulttimeout()

- ii. **settimeout()**
- iii. **getaddrinfo()**
- iv. no such method

18. Which exception is raised for address related errors by getaddrinfo()?

- i. gaoerror
- ii. gsierror
- iii. **gaierror**
- iv. gdeerror

19. _____ protocol facilitates sending of datagrams in an reliable manner.

- i. **TCP**
- ii. UDP
- iii. HTTP
- iv. FTP

20. To create a socket, which function among the following is available in python socket module?

- i. socket.create()
- ii. socket.initialize()
- iii. **socket.socket()**
- iv. socket.build()

Which of the following programming paradigms allow us to write programs and know they are correct before running them?

- a) Automata based Programming Paradigm
- b) Logical Programming Paradigm
- c) **Dependent type Programming Paradigm**
- d) Imperative Programming Paradigm

2) Which of the following is false regarding dependent types?

- a) They allow us to write programs and know they are correct before running them.
- b) **They allow us to write programs and know they are correct only after running them.**
- c) You can specify types that can check the value of your variables at compile time.
- d) Its definition depends on a value.

3) Which of the notations is true for “P(x) is true for all values of x in the universe of discourse”?

- a) $x\forall P(x)$
- b) $\forall P(x)x$
- c) $\exists xP(x)$
- d) **$\forall xP(x)$**

4) Which of the following is false about quantifiers?

- a) Notation \forall is used for Universal quantifier.
- b) Notation \exists is used for Existential quantifier.
- c) “All men drink tea” is an example of Universal Quantifier
- d) **“All men drink coffee” is an example of Existential Quantifier.**

5) Let P(x) be the predicate “x must take a discrete mathematics course” and let Q(x) be the predicate “x is a computer science student”.

Which of the following statements is correct for “Everybody must take a discrete mathematics course or be a computer science student”?

- a) **$\forall x(Q(x) \vee P(x))$**
- b) $\forall x(Q(x)) \vee \forall x(P(x))$
- c) $\forall x(Q(x) \parallel P(x))$
- d) $\forall x(Q(x) \rightarrow P(x))$

6) Which of the following is correct for predicate “All men drink coffee”?

- a) $\forall x \text{men}(x) \rightarrow (x, \text{coffee})$
- b) $\text{drink}(x, \text{coffee}) \rightarrow \forall x \text{men}(x)$

- c) $\forall x \text{ men}(x) \rightarrow \text{drink}(x, \text{coffee})$
d) $\forall \text{men}(x) \rightarrow \text{drink}(x, \text{coffee})$
- 7) Which of the following is correct for predicate "Some boys play football"?
a) $\exists x \text{ boys}(x) \rightarrow \text{play}(x, \text{football})$
b) $\exists \text{ boys}(x) \rightarrow \text{play}(x, \text{football})$
c) $\forall x \text{ boys}(x) \rightarrow \text{play}(x, \text{football})$
d) $\exists x \forall \text{boys}(x) \rightarrow \text{play}(x, \text{football})$
- 8) Dependent Type is used to encode _____ like "for all" and "there exists".
a. **Logic Quantifiers**
b. Analysing Quantifiers
c. Dependent Quantifiers
d. None of the above
- 9) "There exists x in the universe of discourse such that P(X) is true" is which Quantifiers statement?
a. Logical
b. Universal
c. **Existential**
d. None of these
- 10) What is the way to determine if a given function is dependant type
a. Result is independent of the argument
b. Result depends upon the usage in the program
c. **Result depends on the Value of its argument**
d. Result depends on available resources
- 11) Notation for an Existential Quantifier:
a. $\forall x P(x)$
b. $\Sigma P(x)$
c. $\emptyset x P(x)$
d. **$\exists x P(x)$**
- 12) Representation of the following statement:
Every Clock has quartz
a. $\emptyset x \text{clock}(x) \rightarrow \text{quartz}(x)$
b. $\exists x \text{clock}(x) \rightarrow \text{quartz}(x)$
c. **$\forall x \text{clock}(x) \rightarrow \text{quartz}(x)$**
d. none of the above
- 13) Representation of the following statement:
Some leaves are Red

a. $\exists x \text{leaves}(x) \rightarrow \text{red}(x)$

b. $\forall x \text{leaves}(x) \rightarrow \text{red}(x)$

c. $\Sigma \text{leaves}(x) \rightarrow \text{red}(x)$

d. none of the above

13. A function has dependent type if the _____ of a function's result depends on the _____ of its Argument

a. value and type

b. type and value

c. type and type

d. value and value

14. Dependent type paradigm used to encode logic's quantifiers like _____ and _____

a. for one, there may exists

b. for all, there always

c. for all, there exists

d. for specific, there exists

15. Choose the correct one with respect to typing and typing-extensions library in python dependent type programming.

a. typing is not builtin python module where all possible types are defined.

typing_extensions is an official package for new types in the future releases of python

b. typing is a builtin python module where all possible types are defined.

typing_extensions is an official package for new types in the future releases of python

c. typing is a builtin python module where all possible types are defined.

typing_extensions is not an official package for new types in the future releases of python

d. typing is not a builtin python module where all possible types are defined.

typing_extensions is not an official package for new types in the future releases of python

16. Predict the output of the below mentioned python code without dependent type syntax:

```
def make_hamburger(meat, number_of_meats):
    return ["bread"] + [meat] * number_of_meats + ["bread"]
print(make_hamburger("ground beef", 2))
```

a. ['bread', 'ground beef', 2, 'bread']

b. ['bread', 'ground beef', 'ground beef', 'bread']

c. *TypeError: cannot concatenate 'str' and 'int' objects*

d. `['bread', 'ground beef', '2bread']`

17. Predict the output of the below mentioned python code with dependent type syntax:

```
from typing import List

def greet_all(names: List[str]) -> None:
    for name in names:
        print('Hello ' + name)

names = ["Alice", "Bob", "Charlie"]
ages = [10, 20, 30]

greet_all(names)
greet_all(ages)

a. greet_all(names) # Ok!
    greet_all(ages) # Ok!

b. a.greet_all(names) # Ok!
    greet_all(ages) # Error due to incompatible types

c. a.greet_all(names) # Error due to incompatible types
    greet_all(ages) # Ok!

d. a.greet_all(names) # Error due to incompatible types
    greet_all(ages) # Error due to incompatible types
```

18 Let x be a variable which refers to Universe of Disclosure such as x1,x2....xn then

,how to represent this statement using quantifiers “ All Man working in Industry”

- a) $\forall x \text{ man}(x) \rightarrow \text{work}(x, \text{Industry})$ b) $\forall x \text{ man}(x) \rightarrow \text{work}(\text{industry})$.
- a) $\forall x \rightarrow \text{work}(x, \text{industry})$ d) $\forall x \text{ man}(x) \rightarrow \text{Industry}(\text{work})$

19 How do you represent the statement “Every man respects his parent.”

- a) $\forall x \text{ woman}(x) \rightarrow \text{respects}(x, \text{parent})$ b) $\forall x \text{ man}(x) \rightarrow \text{respects}(x, \text{parent})$
- c) $\forall x \text{ man} \rightarrow \text{respects}(x, \text{parent})$ d) $\forall x \text{ man}(x) \rightarrow \text{respects}(\text{parent})$.

20 How do you represent the statement “Some boys play cricket “

- a) $\exists x \text{ boys}(x) \rightarrow \text{play}(\text{all})$ b) $\forall x \text{ boys}(x) \rightarrow \text{play}(x, \text{cricket})$
- c) $\exists x \text{ boys}(x) \rightarrow \text{play}(x, \text{cricket})$ d) $\exists x \wedge \forall x \text{ boys}(x) \rightarrow \text{play}(x, \text{cricket})$

1.Sympy is a collection of mathematical algorithms and convenience functions built on the -----extension of Python

- a) numpy
- b) scikit
- c) sys
- d) functools

2. Exponential function computes the -----

- a) 10^{**x} element-wise
- b) 10^{**x} row-wise
- c) 10^{**x} column-wise
- d) 10^x element-wise

3. _____ evaluates the expression to a floating-point number.

- a) evalf
- b) fval
- c) float
- d) valf

4. what is the output for the following expression $((x+y)^{**2}).expand()$

- a) $x^2 + 2xy + y^2$
- b) $x^2 + 2*x*y + y^2$
- c) $x^2 + 2*x*y + y^2$
- d) $x^{**2} + 2*x*y + y^{**2}$

5. limit($(5^{**x} + 3^{**x})^{**}(1/x)$, x, oo) ,what is the output

- a) 5
- b) oo
- c) 1
- d) 0

6) Higher derivatives can be calculated using the which method

- a) highder(func,var,n)
- b) diff(func, var, n)
- c) diff(n,var,func)
- d) diff(func, var)

7) what is the output

```
>>> x = Symbol('x')
>>> y = Symbol('y')
>>> A = Matrix([[1,x], [y,1]])
>>> A**2
```

- a) $[1, x]$
 $[y, 1]$
- b) $[xy + 1, 2x]$
 $[2y, xy + 1]$
- c) $[x*y + 1, 2*y]$
- d) $[x*y + 1, 2*x]$

[$2*x$, $x*y + 1$]

[$2*y$, $x*y + 1$]

8) .match() method, along with the ----- class, to perform pattern matching on expressions.

- a) pattern
- b) func
- c) wild**
- d) dictionary

9) which among the following function Return or print, respectively, a pretty representation of expr

- a) pretty(expr)
- b) pretty_print(expr)
- c) pprint(expr)
- d) all of the above**

10) What is the output of

```
from sympy.abc import a, b
```

```
expr = b*a + -4*a + b + a*b + 4*a + (a + b)*3
```

- a) $ba-4a+b+ab+4a+3(a+b)$
- b) $2*a*b + 3*a + 4*b$**
- c) $2ab+3a+4b$
- d) all of the above

11) print(pi.evalf(30))

- a) $3.14/30$
- b) $30/3.14$
- c) 3.14159265358979323846264338328**
- d) 3.14

12) which is the correct way to write equation for $x^2=x$ in sympy

- a) $x^{**2} = x$**
- b) $x*x = x$
- c) $x\%2 = x$
- d) $X^2 = x$

13) how to find a solution for a equation in a given interval

- a) solve(equation,range)
- b) equation(solve,range)
- c) solveset()**
- d) both a and b

14) Allows , the same elements can appear multiple times at different positions

a) set **b)sequence**

c) dictionary d) none

15) which is the snippet to find the eigenvalues of $\begin{bmatrix} 1 & 2 \\ 2 & 2 \end{bmatrix}$

- a) `Matrix([[1, 2], [2, 2]]).eigenvals()`
b) `Matrix([[1, 2], [2, 2]]).eigenvalues()`
c) both a and b
d) `eigen([[1, 2], [2, 2]])`

16 What is the purpose of sympify() method?

- a. Convert expression of string type to mathematical expression
- b. Convert mathematical expression to String
- c. Convert mathematical expression to character
- d. Convert tuple to mathematical expression

17 Find the output of the following program

```
from sympy import solve
x = Symbol('x')
expr = x**2 + 5*x + 4
solve(expr, dict=True)
```

- a. $\{x: -4\}, \{x: -1\}$
- b. $\{x: -6\}, \{x: -1\}$
- c. $\{x: -1\}, \{x: -4\}$
- d. $\{x: 4\}, \{x: -1\}$

18. A *rational expression* is an algebraic expression in which the numerator and denominator are both -----

- a. Equal
- b. **Polynomials**
- c. Unequal
- d. Symmetric

19. Find the output of the following program

```
# import sympy
from sympy import *
```

```

x = symbols('x')
expr = sin(x)/x;
print("Expression : {}".format(expr))

# Use sympy.limit() method
limit_expr = limit(expr, x, 0)
print("Limit of the expression tends to 0 : {}".format(limit_expr))

```

- a. Expression : $\cos(x)/x$
Limit of the expression tends to 0 : 2
- b. Expression : $\tan(x)/x$
Limit of the expression tends to 0 : 3
- c. Expression : $\sin(x)/x$
Limit of the expression tends to 1 : 0
- d. **Expression : $\sin(x)/x$**
Limit of the expression tends to 0 : 1

20. ----- method will simplify mathematical expression using trigonometric identities.

- a. **sympy.trigsimp()**
- b. sympy.series()
- c. sympy.lambda()
- d. sympy.sim()

1. What does a threading.Lock do?

- a) Pass messages between threads
- b) Allow only one thread at a time to access a resource**
- c) Wait until a thread is finished
- d) SHIFT TO ALL CAPS

2. What does the Thread.join() method do?

- a) Waits for the thread to finish**
- b) Restricts access to a resource
- c) Adds the thread to a pool
- d) Merges two threads into one

3) Race conditions are ...

- a) Testing which thread completes first
- b) Two threads incorrectly accessing a shared resource**
- c) The weather on race day
- d) Something you should add to your code

4) It sets the lock state to locked. If called on a locked object, it blocks until the resource is free.

- a) lock()
- b)release()
- c) acquire()**
- d)join()

5) You have thread T1, T2, and T3. How will you ensure that thread T2 is run after T1 and thread T3 after T2?

- a) Sleep method
- b) Join Method**
- c)Release
- d)lock method

6) What are the libraries in Python that support threads?

- _threading
- b) Thread
- c)None
- d)thread**

7) Mention the correct syntax for creating Thread object for calling increment methods

- a) t1 = threading.Thread()
- b) t1 = threading.Thread(target)
- c) t1 = threading.Thread(target=incr)**
- d) t1 = threading.Thr

8) Which leads to concurrency?

- a) Serialization
- b) Parallelism**
- c) Serial processing
- d) Distribution

9) Which is not a method for parallelism?

- a) Message Passing
- b) Shared Memory
- c) Threads
- d) Sockets**

10) A race condition occurs when multiple processes or threads read and write

- a). Input b). Information c). **Data Items** d). Programs

11) For a single processor system, implementation of semaphores is possible to inhibited through

- a) Deadlock b) **Interrupts** c) Lock Step d) Paging

12) Which method controls the execution of thread in python?

- a) Wait b) **Sleep** c) Acquire d) Lock

13) How does run() method is invoked?

- a) By Thread.create() b) **By Thread.start()**

- c) None d) By Thread.run()

14) What is the difference between *threading.Lock* and *threading.RLock*?

- a) Lock and RLock both primitives are owned by a single thread.

- b) **Lock is owned by none while RLock is owned by many.**

- c) Lock is owned by a thread while RLock is owned by many.

- d) Lock and RLock both primitives are owned by many.

15) What is the difference between a *semaphore* and *bounded semaphore*?

- a) Semaphore holds a counter for the number of release() calls minus the number of acquire() calls, plus an initial value but bounded semaphore doesn't.

- b) A semaphore makes sure its current value doesn't exceed its initial value while bounded semaphore doesn't.

- c) **A bounded semaphore makes sure its current value doesn't exceed its initial value while semaphore doesn't.**

- d) Bounded semaphore holds a counter for the number of release() calls minus the number of acquire() calls, plus an initial value but semaphore doesn't.

16) What is the method that wakes up all thread waiting for the condition?

- a) releaseAll() b) notify() c) **notifyAll()** d) release()

17 How to terminate a blocking thread?

- a). **thread.stop() & thread.wait()** b) thread.stop()

- c) thread.terminate() d) None

18. Which synchronization method is used to guard the resources with limited capacity, e.g. a database server?

- a) Event b) Condition c) Lock d) **Semaphore**

19. How to detect the status of a python thread?

- a) isActive()
- b) isDaemon()
- c) None
- d) **isAlive()**

20. Execution of several activities at the same time.

- a) processing
- b) parallel processing**
- c) serial processing
- d) multitasking

1. What does a threading.Lock do?

- a) Pass messages between threads **b) Allow only one thread at a time to access a resource**
c) Wait until a thread is finished d) SHIFT TO ALL CAPS

2. What does the Thread.join() method do?

- a) Waits for the thread to finish** b) Restricts access to a resource
c) Adds the thread to a pool d) Merges two threads into one

3) Race conditions are ...

- a) Testing which thread completes first **b) Two threads incorrectly accessing a shared resource**
c) The weather on race day d) Something you should add to your code

4) It sets the lock state to locked. If called on a locked object, it blocks until the resource is free.

- a) lock() b)release() **c) acquire()** d)join()

5) You have thread T1, T2, and T3. How will you ensure that thread T2 is run after T1 and thread T3 after T2?

- a) Sleep method **b) Join Method** c)Release d)lock method

6) What are the libraries in Python that support threads?

- _threading b) Thread c)None **d)thread**

7) Mention the correct syntax for creating Thread object for calling increment methods

- a) t1 = threading.Thread() b) t1 = threading.Thread(target)
c) t1 = threading.Thread(target=incr) d) t1 = threading.Thr

8) Which leads to concurrency?

- a) Serialization **b) Parallelism** c) Serial processing d) Distribution

9) Which is not a method for parallelism?

- a) Message Passing b) Shared Memory c) Threads **d) Sockets**

- 10) A race condition occurs when multiple processes or threads read and write
- a). Input b). Information **c). Data Items** d). Programs
- 11) For a single processor system, implementation of semaphores is possible to inhibited through
- a) Deadlock b) **Interrupts** c) Lock Step d) Paging
- 12) Which method controls the execution of thread in python?
- a) Wait b) **Sleep** c) Acquire d) Lock
- 13) How does run() method is invoked?
- a) By Thread.create() **b) By Thread.start()**
- c) None d) By Thread.run()
- 14) What is the difference between *threading.Lock* and *threading.RLock*?
- a) Lock and RLock both primitives are owned by a single thread.
- b) **Lock is owned by none while RLock is owned by many.**
- c) Lock is owned by a thread while RLock is owned by many.
- d) Lock and RLock both primitives are owned by many.
- 15) What is the difference between a *semaphore* and *bounded semaphore*?
- a) Semaphore holds a counter for the number of release() calls minus the number of acquire() calls, plus an initial value but bounded semaphore doesn't.
- b) A semaphore makes sure its current value doesn't exceed its initial value while bounded semaphore doesn't.
- c) **A bounded semaphore makes sure its current value doesn't exceed its initial value while semaphore doesn't.**
- d) Bounded semaphore holds a counter for the number of release() calls minus the number of acquire() calls, plus an initial value but semaphore doesn't.
- 16) What is the method that wakes up all thread waiting for the condition?
- a) releaseAll() b) notify() c) **notifyAll()** d) release()
- 17 How to terminate a blocking thread?
- a). **thread.stop() & thread.wait()** b) thread.stop()

c) `thread.terminate()` d) None

18. Which synchronization method is used to guard the resources with limited capacity, e.g. a database server?

a) Event b) Condition c) Lock d) **Semaphore**

19. How to detect the status of a python thread?

a) `isActive()` b) `isDaemon()` c) None d) **isAlive()**

20. Execution of several activities at the same time.

a) processing b) **parallel processing** c) serial processing d) multitasking

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

VADAPALANI CAMPUS

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

18CSC207J – ADVANCED PROGRAMMING PRACTICES

NETWORK PROGRAMMING PARADIGM

1. _____ programming is a way of connecting two nodes on a network to communicate with each other.
 - i. Logic
 - ii. **Socket**
 - iii. Functional
 - iv. Symbolic

2. Which method used by the server to initiate the connection with the client?
 - i. Listen()
 - ii. Close()
 - iii. Bind()
 - iv. **Accept()**

3. _____ is a socket through which data can be transmitted continuously.
 - i. Datagram Socket
 - ii. **Stream Socket**
 - iii. Raw Socket
 - iv. Binary Socket

4. _____ paradigm deals with client – server communication
 - i. Dependent type paradigm
 - ii. Parallel programming paradigm
 - iii. Network paradigm
 - iv. Concurrent programming paradigm

5. _____ protocol facilitates sending of datagrams in an unreliable manner.

- i. TCP
 - ii. UDP**
 - iii. HTTP
 - iv. FTP
6. Which among methods are not server socket?
- i. connect()**
 - ii. bind()
 - iii. listen()
 - iv. accept()
7. In UDP, Which among methods are used to receive messages at endpoint
- i. sock_object.recv()
 - ii. sock_object.send()
 - iii. sock_object.recvfrom()**
 - iv. sock_object.sendto()
8. In TCP, Which among methods are used to send messages from endpoint
- i. sock_object.recv()
 - ii. sock_object.send()**
 - iii. sock_object.recvfrom()
 - iv. sock_object.sendto()
9. The protocol which defines IPv4 is
- i. AF_UNIX
 - ii. SOCK_STREAM
 - iii. AF_INET**
 - iv. SOCK_DGRAM
10. The correct order of methods used in server socket is
- i. Socket(), Bind(), listen(), accept()**
 - ii. Socket(), listen(), bind(), accept()
 - iii. Socket(), accept(), bind(), listen()
 - iv. Socket(), bind(), accept(), listen()

11. _____ is a type of network socket which provides connection less point for sending and receiving packets.

- i. **Datagram Socket**
- ii. Stream Socket
- iii. Raw Socket
- iv. Binary Socket

12. _____ do not use any transport protocol but data is directly transmitted over IP protocol

- i. Datagram Socket
- ii. Stream Socket
- iii. **Raw Socket**
- iv. Binary Socket

13. The _____ is a physical path over which the message travels.

- i. Protocol
- ii. **Medium**
- iii. Path
- iv. Route

14. A pair (host, port) is used for the _____ address family.

- i. AF_NETLINK
- ii. AF_INET6
- iii. **AF_INET**
- iv. AF_ALG

15. Use _____ to make the socket to visible to the outside world.

- i. socket.listen()
- ii. socket.visible()
- iii. socket.socket()
- iv. **Socket.gethostname()**

16. In which mode socket is created in default.

- i. **Blocking mode**
- ii. Non-Blocking Mode
- iii. Timeout mode
- iv. Accept mode

17. Which method is recommended to be called before calling connect() method?

- i. getdefaulttimeout()

- ii. **settimeout()**
- iii. **getaddrinfo()**
- iv. no such method

18. Which exception is raised for address related errors by getaddrinfo()?

- i. gaoerror
- ii. gsierror
- iii. **gaierror**
- iv. gdeerror

19. _____ protocol facilitates sending of datagrams in an reliable manner.

- i. **TCP**
- ii. UDP
- iii. HTTP
- iv. FTP

20. To create a socket, which function among the following is available in python socket module?

- i. socket.create()
- ii. socket.initialize()
- iii. **socket.socket()**
- iv. socket.build()

SYMBOLIC PARADIGM

ANNOTATION

PYTHON CODE	MATH EXPRESSION	PYTHON OUTPUT
#Using Math Library import math print(math.sqrt(8))	$\sqrt{8}$	2.8284271247461903
#using Sympy Library import sympy print(sympy.sqrt(8))	$\sqrt{8} = 2*2*2 = 2 * \sqrt{2}$	$2*\sqrt{2}$
#using symbols from sympy import symbols x=symbols('x') y=symbols('y') print(x+y+x-y) print((x+y)*(x+y))	$2X$ $(X+Y)(X+Y) = (X+Y)^2$	$2*x$ $(x + y)^{**2}$
#using expand from sympy import expand exp = expand((x+y)**3) print(exp)	$(x+y)^3 = x^3+3x^2y+3xy^2+y^3$	$x^{**3} + 3*x^{**2}*y + 3*x*y^{**2} + y^{**3}$
#using factor from sympy import factor exp = expand((x+y)**3)	Factor($x^3+3x^2y+3xy^2+y^3$) $= (x+y)^3$	$(x + y)^{**3}$

print(factor(exp))		
#using simplify from sympy import simplify print(simplify((x+x*y)/x))	$\frac{xy + x}{x} = y + 1$	y + 1
#using limit and sine from sympy import limit,sin print(limit(sin(x)/x,x,0))	$\lim_{x \rightarrow 0} \frac{\sin(x)}{x} = 1$	1
#using infinity from sympy import oo print(limit(x,x,oo))	$\lim_{x \rightarrow \infty} x = \infty$	00
#using diff from sympy import diff,tan print(diff(sin(x),x)) print(diff(sin(2*x),x)) print(diff(tan(x),x)) print(diff(sin(2*x),x,1))	$\frac{d}{dx}(\sin(x)) = \cos(x)$ $\frac{d}{dx}(\sin(2x)) = 2\cos(2x)$ $\frac{d}{dx}(\tan(x)) = \sec^2(x)$ $\frac{\partial \sin(2x)}{\partial x} \text{ where } x = 1 = 2\cos(2)$	$\cos(x)$ $2*\cos(2*x)$ $\tan(x)^{*}2 + 1$ $2*\cos(2*x)$
#using series from sympy import series,cos print(series(cos(x),x))	$1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{x^6}{720} + O(x^7)$ <p>(Taylor series)</p>	$1 - x^{*}2/2 + x^{*}4/24 + O(x^{*}6)$
#using integration from sympy import integrate print(integrate(6*x **5,x)) print(integrate(sin(x),x)) print(integrate(x**3, (x,-1,1)))	$\int 6x^5 dx = x^6 + \text{constant}$ $\int \sin(x) dx = -\cos(x) + \text{constant}$	$x^{*}6$ $-\cos(x)$

	$\int_{-1}^1 x^3 dx = 0$	0
#equation solving from sympy import solve print(solve((x+5*y-2,-3*x+6*y-15),(x,y)))	X + 5y - 2 -3x+6y-15 $x = -3, \quad y = 1$	{x: -3, y: 1}
#matrix from sympy import Matrix x=Matrix([[1,2],[3,4]]) y=Matrix([[1,3],[5,6]]) print(x+y) print(x-y) print(x*y)	$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} 1 & 3 \\ 5 & 6 \end{pmatrix} = \begin{pmatrix} 2 & 5 \\ 8 & 10 \end{pmatrix}$ $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} - \begin{pmatrix} 1 & 3 \\ 5 & 6 \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ -2 & -2 \end{pmatrix}$ $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \cdot \begin{pmatrix} 1 & 3 \\ 5 & 6 \end{pmatrix} = \begin{pmatrix} 11 & 15 \\ 23 & 33 \end{pmatrix}$ Matrix([[2, 5], [8, 10]]) Matrix([[0, -1], [-2, -2]]) Matrix([[11, 15], [23, 33]])
#using Rational import sympy a = sympy.Rational(1, 2) print(a)	$\frac{1}{2}$	1/2
#using pi import sympy print(sympy.pi) print(sympy.pi**2) print(sympy.pi.evalf()) print((sympy.pi+sympy.exp(1)).evalf())	π $\pi + e$	pi pi**2 3.14159265358979 5.85987448204884
#using solveset from sympy import solveset print(solveset(x**2 + 1, x)) print(solveset(x ** 4 - 1, x))	$x^2 + 1 = 0 \Rightarrow x = \pm i$	{-I, I} {-1, 1, -I, I}

$$\boxed{x^4 - 1 = 0 \Rightarrow \begin{array}{c} x = \pm 1 \\ \hline x = \pm i \end{array}}$$

Which of the following programming paradigms allow us to write programs and know they are correct before running them?

- a) Automata based Programming Paradigm
- b) Logical Programming Paradigm
- c) **Dependent type Programming Paradigm**
- d) Imperative Programming Paradigm

2) Which of the following is false regarding dependent types?

- a) They allow us to write programs and know they are correct before running them.
- b) **They allow us to write programs and know they are correct only after running them.**
- c) You can specify types that can check the value of your variables at compile time.
- d) Its definition depends on a value.

3) Which of the notations is true for “P(x) is true for all values of x in the universe of discourse”?

- a) $\forall x P(x)$
- b) $\exists P(x)x$
- c) $\exists x P(x)$
- d) **$\forall x P(x)$**

4) Which of the following is false about quantifiers?

- a) Notation \forall is used for Universal quantifier.
- b) Notation \exists is used for Existential quantifier.
- c) “All men drink tea” is an example of Universal Quantifier
- d) **“All men drink coffee” is an example of Existential Quantifier.**

5) Let P(x) be the predicate “x must take a discrete mathematics course” and let Q(x) be the predicate “x is a computer science student”.

Which of the following statements is correct for “Everybody must take a discrete mathematics course or be a computer science student”?

- a) **$\exists x(Q(x) \vee P(x))$**
- b) $\exists x(Q(x)) \vee \exists x(P(x))$
- c) $\exists x(Q(x) \parallel P(x))$
- d) $\exists x(Q(x) \rightarrow P(x))$

6) Which of the following is correct for predicate “All men drink coffee”?

- a) $\exists x \text{ men}(x) \rightarrow (x, \text{coffee})$
- b) drink(x, coffee) $\rightarrow \exists x \text{ men}(x)$
- c) $\exists x \text{ men}(x) \rightarrow \text{drink}(x, \text{coffee})$**
- d) $\exists \text{men}(x) \rightarrow \text{drink}(x, \text{coffee})$
- 7) Which of the following is correct for predicate "Some boys play football"?
- a) **$\exists x \text{ boys}(x) \rightarrow \text{play}(x, \text{football})$**
- b) $\exists \text{boys}(x) \rightarrow \text{play}(x, \text{football})$
- c) $\exists x \text{ boys}(x) \rightarrow \text{play}(x, \text{football})$
- d) $\exists x \exists \text{boys}(x) \rightarrow \text{play}(x, \text{football})$
- 8) Dependent Type is used to encode _____ like "for all" and "there exists".
- Logic Quantifiers**
 - Analysing Quantifiers
 - Dependent Quantifiers
 - None of the above
- 9) "There exists x in the universe of discourse such that P(X) is true" is which Quantifiers statement?
- Logical
 - Universal
 - Existential**
 - None of these
- 10) What is the way to determine if a given function is dependant type
- Result is independent of the argument
 - Result depends upon the usage in the program
 - Result depends on the Value of its argument**
 - Result depends on available resources
- 11) Notation for an Existential Quantifier:
- $\exists x P(x)$
 - $\Sigma P(x)$
 - $\exists x P(x)$
 - $\exists x P(x)$**
- 12) Representation of the following statement:
Every Clock has quartz
- $\exists x \text{clock}(x) \rightarrow \text{quartz}(x)$
 - $\exists x \text{clock}(x) \rightarrow \text{quartz}(x)$
 - $\exists x \text{clock}(x) \rightarrow \text{quartz}(x)$**

d. none of the above

13) Representation of the following statement:

Some leaves are Red

a. ~~$\exists x \text{leaves}(x) \rightarrow \text{red}(x)$~~

b. $\exists x \text{leaves}(x) \rightarrow \text{red}(x)$

c. $\Sigma \text{leaves}(x) \rightarrow \text{red}(x)$

d. none of the above

13. A function has dependent type if the _____ of a function's result depends on the _____ of its Argument

a. value and type

b. type and value

c. type and type

d. value and value

14. Dependent type paradigm used to encode logic's quantifiers like _____ and _____

a. for one, there may exists

b. for all, there always

c. for all, there exists

d. for specific, there exists

15. Choose the correct one with respect to typing and typing-extensions library in python dependent type programming.

a. typing is not builtin python module where all possible types are defined.

~~typing_extensions is an official package for new types in the future releases of python~~

b. typing is a builtin python module where all possible types are defined.

~~typing_extensions is an official package for new types in the future releases of python~~

c. typing is a builtin python module where all possible types are defined.

~~typing_extensions is not an official package for new types in the future releases of python~~

d. typing is not a builtin python module where all possible types are defined.

~~typing_extensions is not an official package for new types in the future releases of python~~

16. Predict the output of the below mentioned python code without dependent type syntax:

```
def make_hamburger(meat, number_of_meats):
    return ["bread"] + [meat] * number_of_meats + ["bread"]
```

```
print(make_hamburger("ground beef", 2))
```

- a. ['bread', 'ground beef', 2, 'bread']
- b. ['bread', 'ground beef', 'ground beef', 'bread']
- c. *TypeError: cannot concatenate 'str' and 'int' objects*
- d. ['bread', 'ground beef', '2bread']

17. Predict the output of the below mentioned python code with dependent type syntax:

```
from typing import List

def greet_all(names: List[str]) -> None:
    for name in names:
        print('Hello ' + name)

names = ["Alice", "Bob", "Charlie"]
ages = [10, 20, 30]

greet_all(names)
greet_all(ages)

a.greet_all(names)    # Ok!
greet_all(ages)      # Ok!

b. a.greet_all(names)    # Ok!
greet_all(ages)      # Error due to incompatible types

c. a.greet_all(names)  # Error due to incompatible types
greet_all(ages)      # Ok!

d. a.greet_all(names)  # Error due to incompatible types
greet_all(ages)      # Error due to incompatible types
```

18 Let x be a variable which refers to Universe of Disclosure such as x1,x2....xn then how to represent this statement using quantifiers “ All Man working in Industry”

- a) $\exists x \text{ man}(x) \rightarrow \text{work}(x, \text{Industry})$ b) $\exists x \text{ man}(x) \rightarrow \text{work}(\text{industry})$.
- a) $\exists x \rightarrow \text{work}(x, \text{industry})$ d) $\exists x \text{ man}(x) \rightarrow \text{Industry}(\text{work})$

19 How do you represent the statement “Every man respects his parent.”

- a) $\exists x \text{ woman}(x) \rightarrow \text{respects}(x, \text{parent})$ b) $\exists x \text{ man}(x) \rightarrow \text{respects}(x, \text{parent})$
- c) $\exists x \text{ man} \rightarrow \text{respects}(x, \text{parent})$ d) $\exists x \text{ man}(x) \rightarrow \text{respects}(\text{parent})$.

20 How do you represent the statement “Some boys play cricket “

- a) $\exists x \text{ boys}(x) \rightarrow \text{play}(\text{all})$ b) $\exists x \text{ boys}(x) \rightarrow \text{play}(x, \text{cricket})$

c) $\exists x \text{ boys}(x) \rightarrow \text{play}(x, \text{cricket})$

d) $\exists x^{\wedge} \exists x \text{ boys}(x) \rightarrow \text{play}(x, \text{cricket})$

1.Sympy is a collection of mathematical algorithms and convenience functions built on the ----- extension of Python

a) numpy
c) sys

b) scikit
d) functools

2. Exponential function computes the -----

a) 10^{**x} element-wise
c) 10^{**x} column-wise

b) 10^{**x} row-wise
d) 10^x element-wise

3. _____ evaluates the expression to a floating-point number.

a) evalf
c) float

b) fval
d) vvalf

4. what is the output for the following expression $((x+y)^{**2}).expand()$

a) $x^2 + 2xy + y^2$
c) $x^2 + 2xy + y^2$

b) $x^2 + 2*x*y + y^2$
d) $x^{2} + 2*x*y + y^{**2}$**

5. limit($(5^{**x} + 3^{**x})^{**}(1/x)$, x, oo) ,what is the output

a) 5
c) 1

b) oo
d) 0

6) Higher derivatives can be calculated using the which method

a) highder(func,var,n)
c) diff(n,var,func)

b) diff(func, var, n)
d) diff(func, var)

7) what is the output

```
>>> x = Symbol('x')  
>>> y = Symbol('y')  
>>> A = Matrix([[1,x], [y,1]])  
>>> A**2
```

a) $[1, x]$
b) $[xy + 1, 2x]$

[y, 1]

[2y, xy + 1]

c) $[x^*y + 1, 2^*y]$
[$2^*x, x^*y + 1$]

d) $[x^*y + 1, 2^*x]$
[$2^*y, x^*y + 1$]

8) .match() method, along with the ----- class, to perform pattern matching on expressions.

a) pattern

b) func

c) **wild**

d) dictionary

9) which among the following function Return or print, respectively, a pretty representation of expr

a) pretty(expr)

b) pretty_print(expr)

c) pprint(expr)

d) **all of the above**

10) What is the output of

```
from sympy.abc import a, b
```

```
expr = b*a + -4*a + b + a*b + 4*a + (a + b)*3
```

a) $ba-4a+b+ab+4a+3(a+b)$

b) **$2^*a^*b + 3^*a + 4^*b$**

c) $2ab+3a+4b$

d) all of the above

11) print(pi.evalf(30))

a) 3.14/30

b) 30/3.14

c) **3.14159265358979323846264338328**

d) 3.14

12) which is the correct way to write equation for $x^2=x$ in sympy

a) **$x^{**2} = x$**

b) $x*x = x$

c) $x\%2 = x$

d) $X^2 = x$

13) how to find a solution for a equation in a given interval

a) solve(equation,range)

b) equation(solve,range)

c) solveset()

d) both a and b

14) Allows , the same elements can appear multiple times at different positions

a) set

b)sequence

c) dictionary

d) none

15) which is the snippet to find the eigenvalues of $\begin{bmatrix} 1 & 2 \\ 2 & 2 \end{bmatrix}$

a) Matrix([[1, 2], [2, 2]]).eigenvals()

b) Matrix([[1, 2], [2, 2]]).eigen

c) both a and b

d) eigen(([1, 2], [2, 2]))

16 What is the purpose of sympify() method?

- Convert expression of string type to mathematical expression
- Convert mathematical expression to String
- Convert mathematical expression to character
- Convert tuple to mathematical expression

17 Find the output of the following program

```
from sympy import solve
x = Symbol('x')
expr = x**2 + 5*x + 4
solve(expr, dict=True)
```

- [{x: -4}, {x: -1}]
- [{x: -6}, {x: -1}]
- [{x: -1}, {x: -4}]
- [{x: 4}, {x: -1}]

18. A *rational expression* is an algebraic expression in which the numerator and denominator are both -----

- a. Equal
- b. **Polynomials**
- c. Unequal
- d. Symmetric

19. Find the output of the following program

```
# import sympy
from sympy import *
x = symbols('x')
expr = sin(x)/x;
print("Expression : {}".format(expr))

# Use sympy.limit() method
limit_expr = limit(expr, x, 0)
print("Limit of the expression tends to 0 : {}".format(limit_expr))
```

- a. Expression : $\cos(x)/x$
Limit of the expression tends to 0 : 2
- b. Expression : $\tan(x)/x$
Limit of the expression tends to 0 : 3
- c. Expression : $\sin(x)/x$
Limit of the expression tends to 1 : 0
- d. **Expression : $\sin(x)/x$**
Limit of the expression tends to 0 : 1

20. ----- method will simplify mathematical expression using trigonometric identities.

- a. **sympy.trigsimp()**
- b. sympy.series()
- c. sympy.lambda()
- d. sympy.sim()

UNIT V

PART A: MULTIPLE CHOICE QUESTIONS

Symbolic:

1. Which of the following is false about sympy? (CLO5-L1)
 - a. Sympy is a python library for symbolic mathematics
 - b. It requires external libraries for execution
 - c. It is an alternative to the systems like mathematica or maple

Ans: B

2. Symbols can now be manipulated using some of python operators using _____.(CLO5-L1)
A)+. B) && C) ? D\$

Ans: A

3. Which of the following belongs to the numerical type of sympy (CLO5-L1)
 - a. Float
 - b. Integer
 - c. Decimal
 - d. Factorial

Ans: B

4. Choose the correct output for the following code?(CLO5-L2)

```
Import sympy as sym
```

```
a= sym.Rational(4,6)
```

```
print a
```

a. 6/4

b. 0.66

c. 4/6

d. 1.5

Ans: C

5. What is the output for the trigonometry function? (CLO5-L2)

```
Sym.expand(sym.cos(x+y),trig=true)
```

a. sin(x)*sin(y)+cos(x)*cos(y)

b. sin(x)*cos(y)+cos(x)*sin(y)

c. -sin(x)*sin(y) - cos(x)*cos(y)

d. -sin(x)*sin(y) + cos(x)*cos(y)

Ans: D

6. Differentiate the Sympy Expression using the syntax (CLO5-L1)

a. diff (var,func)

b. diff(func,var)

c. diff(expr,var)

d. diff(var,point)

Ans: B

7. SymPy is able to solve algebraic equations, in one and several variables using(CLO5-L1)

- a. solveset()
- b. series()
- c. real()
- d. limit()

Ans: A

8. Which of the following belongs to the calculus function? (CLO5-L1)

- i. Limit
 - ii. Series
 - iii. Arithmetic
 - iv. Computation
- a. Both i,ii
 - b. Both ii,iii
 - c. Both ii,iv
 - d. Both i,iii,iv

Ans: A

9. Choose the output for the following code? (CLO5-L2)

`Limit (sin(x), x,0)`

- a. 0
- b. 1
- c. Infinite
- d. Error

Ans: B

10. Which of the following is the correct output for the below given code? (CLO5-L2)

`Sym.integrate(x**3, (x,-1,1))`

- a. 1
- b. 3
- c. 0
- d. -1

Ans: C

11. Which of the following is the correct output for the below given code? (CLO5-L2)

`x,y=sym.symbols('x,y')`

`A=sym.Matrix([[1,x],[y,1]])`

`Print A`

- a. Matrix ([[1,x], [y,1]])
- b. Matrix ([[x,1],[1,y]])
- c. Matrix ([[0,x], [y,0]])
- d. Matrix ([[x,0],[0,y]])

Ans: A

12. Choose the output for the following code? (CLO5-L2)

`Sym.solveset(x**4-1,x)`

- a. {1,-1,I,-I}
- b. {-1,1,-I,I}
- c. {0,1,I,-I}
- d. {1,-1,-I,I}

Ans: B

13. Limit the Sympy Expression using the syntax (CLO5-L1)
- a. limit (var,func,point)
 - b. limit(func,var,point)
 - c. limit(func,var)
 - d. limit(var,point)

Ans: B

14. Finite state machines are used for____(CLO5-L1)

- a. Pseudo random test patterns
- b. Deterministic test patterns
- c. Random test patterns
- d. Algorithmic test patterns

Ans:D

15. According to the given transitions, which among the following are the epsilon closures of q1 for the given NFA? (CLO5-L3)

$$\Delta(q_1, \varepsilon) = \{q_2, q_3, q_4\}$$

$$\Delta(q_4, 1) = q_1$$

$$\Delta(q_1, \varepsilon) = q_1$$

- a. q4
- b. q2
- c. q1
- d. q1, q2, q3, q4

Ans: D

16. Which of the following tuples order are correct for Deterministic Finite Automata (DFA) (CLO5-L1)

- a. (Q, S, d, q_0, F)
- b. (Q, S, d, F, q_0)
- c. (S, Q, d, q_0, F)
- d. (Q, S, q_0, d, F)

Ans: A

17. NFA, is called as 'non-deterministic' because of :(CLO5-L1)

- a. The result is undetermined
- b. The choice of path is non-deterministic
- c. The state to be transited next is non-deterministic
- d. All of the mentioned

Ans: B

18. _____ is a class attribute defined by its source state and destination state. .(CLO5-L1)

- a. LGPL
- b. Scipy
- c. Transition
- d. State

Ans : C

19. Among the following which is used to define the behavior of what we want to achieve (CLO5-L1)

- a. State
- b. Automata
- c. Transition
- d. Machine

Ans: D

20. _____is a triplet consisting of two states and one command needed for the change of one state to the other. .(CLO5-L1)

- a. machine
- b. Automata
- c. State
- d. Transitions

Ans:D

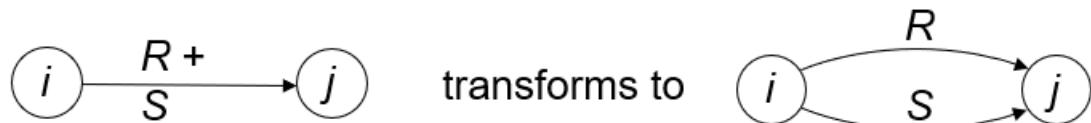
21. _____is a mathematical model of computation. .(CLO5-L1)

- a. state machine
- b. Automata
- c. State
- d. Transitions

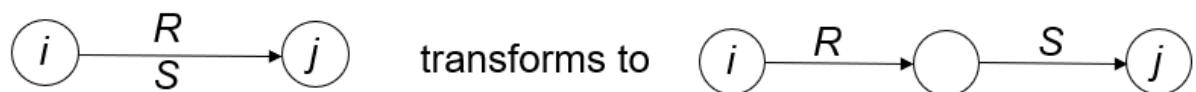
Ans: A

22. Which of the following is correct among the following expressions? (CLO5-L2)

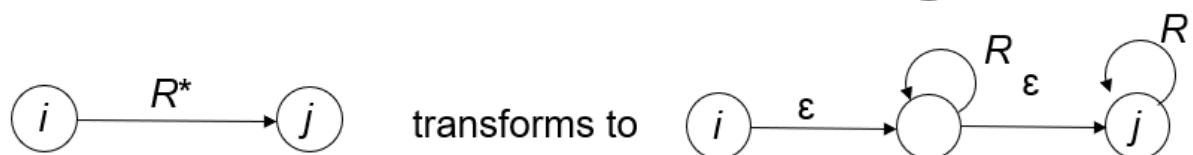
i.



ii.



iii.



- a. Both i,ii
- b. Both ii,iii
- c. Both i,iii
- d. All of the above

Ans: A

23. What kind of abstract machine can recognize strings in a regular set? (CLO5-L1)

- a. DFA
- b. NFA
- c. PDA
- d. None of the above

Ans: A

24. Which of the following statements is wrong? (CLO5-L1)

- a. The language accepted by finite automata are the languages denoted by regular expressions
- b. For every DFA there is a regular expression denoting its language
- c. For a regular expression r , there does not exist NFA with $L(r)$ any transit that accept
- d. None of the above.

Ans: C

25. Let for $\Sigma = \{0,1\}$ $R = (\Sigma\Sigma\Sigma)^*$, the language of R would be-----
(CLO5-L2)

- a. $\{w \mid w \text{ is a string of odd length}$
- b. $\{w \mid w \text{ is a string of length multiple of 3}\}$
- c. $\{w \mid w \text{ is a string of length 3}\}$
- d. All of the above.

Ans: B

26. $(a+b)^*$ is equivalent to----- (CLO5-L2)

- a. b^*a^*
- b. $(a^*b^*)^*$
- c. a^*b^*
- d. None of the above.

Ans: B

27. In regular expressions, the operator '*' stands for----- (CLO5-L1)

- a. Concatenation
- b. Addition
- c. Selection
- d. Iteration

Ans: D

28. The regular expression with all strings of 0's and 1's with at least two consecutive 0's is----- (CLO5-L2)

- a. $1 + (10)^*$ yh
- b. $(0+1)^*011$
- c. $(0+1)^*00(0+1)^*$
- d. $0^*1^*2^*$

Ans: C

29. Which of the following operation can be applied on regular expressions? (CLO5-L1)

- a. Union
- b. Concatenation
- c. Closure
- d. All of the above

Ans: D

30. Finite state machine will initially set to all zeroes. (CLO5-L1)

- a. True
- b. False

Ans: A

31. _____ allow millions of different machines, using all sorts of different network hardware, to pass packets to each other over the fabric of an IP network.(CLO5-L1)
a)Sockets

- b)Client machine
- c)server machines
- d)IP address

Ans: d

32. Which are all necessary to direct a packet to its destination.(CLO5-L2)

- a)IP Address
- b)IP address and port
- c)port
- d)InternetAssignedNumbers

Ans:b

33. _____ (0–1023) are for the most important and widely-used protocols(CLO5-L1)

- a. Well Known ports
- b)Registered ports
- c)Known ports
- d)Unregistered ports

Ans:a

34. _____ (1024–49151) are not usually treated as special by operating systems(CLO5-L1)

- a)Registered ports
- b)well known ports
- c)Known ports
- d)Unregistered ports

Ans:a

35. Python's standard socket module supports _____(CLO5-L1)

- a)getByName()
- b)getName()
- c)gethostbyname()
- d)getServName()

Ans:c

36. Example for Connection oriented protocol(CLO5-L1)

- a)UDP
- b)SMTP
- c)FTP
- d)TCP

Ans:d

37. Example for Connection less protocol(CLO5-L1)

- a)SMTP
- b)FTP
- c)UDP
- d)TCP

Ans:c

38. _____ is an application-level block of transmitted data.(CLO5-L1)

- a)data
- b)datagram
- c)segmentation
- d)Fragmentation

Ans:b

39. Simple server uses _____ command to request a UDP network address.(CLO5-L1)

- a)bind()
- b)socket()
- c)getName()
- d)getsockName()

Ans:a

40. Which method is used by python program to retrieve the current IP and port to which the socket is bound. (CLO5-L2)

- a)bind()
- b)getsockname()
- c)getName()
- d)getSocket()

Ans:b

41. Sending packets with another computer's return address is called _____(CLO5-L1)

- a)Binding
- b) Unification
- c)Spoofing
- d) IP address

Ans:c

42. The concept by which larger UDP packets are spitted into several small physical packets(CLO5-L1)

- a)fragmentation
- b)Binding
- c) Unification
- d) partition

Ans:a

43. The MTU is _____ that all of the network devices between two hosts will support. (CLO5-L1)

- a)smallest packet size
- b)medium packet size
- c)average packet size
- d)largest packet size

Ans:d

44. _____ is efficient only if your host ever only sends one message at a time, then waits for a response.(CLO5-L1)

- a)UDP
- b)TCP
- c)FTP
- d)SMTP

Ans: a

45. Which protocol provides reliable connection?(CLO5-L2)

- a)UDP
- b) FTP
- c)TCP
- d)SMTP

Ans:c

46. TCP uses a _____ that counts the number of bytes transmitted.(CLO5-L1)

- a) protocol
- b)counter
- c)rules
- d)Ports

Ans:b

47. The amount of data that a sender is willing to have on the wire at any given moment is called _____(CLO5-L1)
a) The size of the TCP window. b) counter c) port d) protocol

Ans:a

48. TCP uses _____ to distinguish different applications running at the same IP address, and follows exactly the same conventions regarding well-known and ephemeral port numbers.(CLO5-L1)
a) IP address b)port c)port numbers d) socket ID
Ans:c

49. _____refers to the address family ipv4.(CLO5-L1 (CLO5-L1))
a) Af_INET b) AS_INET c) AG_INET d) AN_INET

Ans:a

50. A server has a _____method which puts the server into listen mode.(CLO5-L1)
a)list() b)listento() c)listen() d)listenfrom()

Ans:c

51. _____ are the end-point of a two-way communication .(CLO5 L1) a)IP Address b)sockets c)sockets ID d) Object ID
Ans:b

52. Identify the method used to connect the client to host and port and initiate the connection towards the server.(CLO5-L3)
a)sock_object.recv() b)sock_object.send()
c)sock_object.append() d)sock_object.connect():
Ans:d

53. Identify the method to receive messages at endpoints when the value of the protocol parameter is TCP.(CLO5-L3)
a)sock_object.recv() b)sock_object.send()
c)sock_object.append() d)sock_object.connect():
Ans:a

54. Which method is used to receive messages at endpoints if the protocol used is UDP.(CLO5-L2)
a)sock_object.recv() b)sock_object.recvfrom()
c)sock_object.append() d)sock_object.connect():
Ans:b

55. Identify the method that returns host name.(CLO5-L1)

- a)sock_object.recv() b)sock_object.recvfrom()
c)sock_object.gethostname d)sock_object.connect():
Ans:c

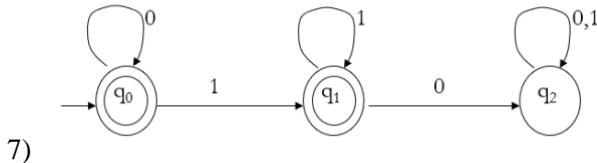
56. Identify the method to send messages from endpoints if the protocol parameter is UDP.(CLO5-L3)
a)sock_object.sendto() b)sock_object.recvfrom()
c)sock_object.gethostname d)sock_object.connect():
Ans:a

57. _____can be used to end direction of communication in a socket(CLO5-L1)
a. The shutdown() call b)Shut() c)close() d)end()

Ans:c

PART: B (4MARKS)

- 1) Write a program to factorize the following expression. (CLO5-L3)
 $x^{**3} + 3*x^{**2}*y + 3*x*y^{**2} + y^{**3}$
- 2) What is SymPy? (CLO5-L1)
- 3) Write the commands to perform the operations on substitutions and expressions (CLO5-L1)
- 4) Design a DFA that accepts all strings that contain 010 or do not contain 0. (CLO5-L2)
- 5) Differentiate Finite state machine and Non deterministic Finite state machine. (CLO5-L2)
- 6) Let $\Sigma = \{0, 1\}$. Give DFAs for $\{\}$, $\{\epsilon\}$, Σ^* , and Σ^+ . (CLO5-L2)



- 7) Provide the DFA function for the above-mentioned diagram (CLO5-L2)

- 8) Compare the features of LISP and Wolfram. (CLO5-L2)
- 9) Write a DFA automata code for $L(M) = \{ w \mid w \text{ has an even number of } 1s \}$ (CLO5-L2)
- 10) Write a DFA automata code for $L(M) = \{(ab)^n \mid n \in \mathbb{N}\}$ (CLO5-L2)
- 11) Construct NFA that recognizes the language of strings that end in 01 (CLO5-L2)
- 12) Find an NFA to recognize the language $(a + ba)^*bb(a + ab)^*$. (CLO5-L2)
- 13) Write the features of GUI programming. (CLO5-L1)
- 14) Explain briefly Automata based programming in python. (CLO5-L1)
- 15) Write the syntax for Expand and Factor command. Give example. (CLO5-L1)
- 16) Give example for DFA and explain. (CLO5-L2)
- 17) Differentiate Python and Jython. (CLO5-L2)
- 18) Write the syntax for series and Integration command. Give example .(CLO5-L1)
- 19) Give example for NFA and explain. (CLO5-L1)
- 20) Differentiate TCP and UDP(CLO5-L2)
- 21) Compare connection oriented and connectionless service(CLO5-L2)
- 22) Differentiate automatic and manual configuration(CLO5-L2)
- 23) How will you generate random port numbers?(CLO5-L2)
- 24) What is file descriptors? Give example(CLO5-L1)
- 25) Differentiate bind() and getSockName() .(CLO5-L2)
- 26) What is UDP fragmentation? Give example .(CLO5-L1)
- 27) What will happen when you send large packets? Give example.(CLO5-L2)
- 28) Differentiate multicast and broadcast.(CLO5-L2)
- 29) Write the code to connect server and client in TCP.(CLO5-L3)
- 30) How does TCP provides reliable connection.(CLO5-L2)

- 31) Compare passive and active socket.(CLO5-L2)
 32) Write the code to connect server and client in UDP.(CLO5-L3)

PART-C (12 MARKS)

1. Write a program to expand and factorize the following expression. (CLO5-L3)

- a. $x^3 + 3x^2y + 3xy^2 + y^3 = (x + y)^3$
- b. $x + y + x^*y$

2. Consider the following series:

$$X + (X^2/2) + (X^3/3) + (X^4/4) + \dots + (X^n/n)$$

Write a python program that will ask a user to input a number, n, and print this series for that number. In the series, x is a symbol and n are an integer input by the program's user. The nth term in this series is given as (X^n/n) . (CLO5-L3)

- 3. Write a program to implement client server communication using UDP. (CLO5-L3)
 - 4. Write a program to demonstrate the concept of UDP fragmentation and explain .(CLO5-L3)
 - 5. Explain automata-based programming paradigm. (CLO5-L1)
 - 6. Design go, slowdown and stop events according to the traffic scenario to cross the road using python code and also draw transition diagram and transition table for the same scenario. (CLO5-L3)
 - 7. Write a example program to find limits, differentiation, Series and Integration. (CLO5-L1)
 - 8. Differentiate DFA and NFA. Explain NFA with example. (CLO5-L2)
 - 9. Build an automaton that accepts all and only those strings that contain 001 (CLO5-L2)
 - 10. Find an DFA for each of the following languages over the alphabet {a, b}.(CLO5-L3)
 - (a) $\{(ab)^n \mid n \in \mathbb{N}\}$, which has regular expression $(ab)^*$.
 - b) Find a DFA for the language of $a + aa^*b$.
 - 11.a. Write NFA automata code for the Language that accepts all end with 01 (CLO5-L3)
 - b. Write a automata code for $L(M) = a + aa^*b + a^*b$.
 - c Write a automata code for Let $\Sigma = \{0, 1\}$.
- Given NFAs for $\{\}$, $\{\epsilon\}$, $\{(ab)^n \mid n \in \mathbb{N}\}$, which has regular expression $(ab)^*$.
12. Write a program to convert NFA to DFA (CLO5-L2)
13. Explain in detail about communication using UDP with example. .(CLO5-L1)

UNIT 4

Dependent Programming paradigm, parallel and concurrent Programming paradigm

1. Parallelism representation is critical to the success of ----- (CLO-4,L1)

- a)High-performance computing.
- b)Low-performance computing
- c)Scaling
- d)Vectorization

Ans:a

2. Parallel programming through a combination of -----and ----- (CLO-4,L1)

- a.Patterns, examples
- b.Algorithms , flowcharts
- c.Models , methods
- d.Classes ,objects

Ans:a

3.What is multithreaded programming? (CLO-4,L1)

- a) It's a process in which two different processes run simultaneously
- b) It's a process in which two or more parts of same process run simultaneously
- c) It's a process in which many different process are able to access same information
- d) It's a process in which a single process can access information from many sources

Ans:b

4. Which of these are types of multitasking? (CLO-4,L2)

- a) Process based
- b) Thread based
- c) Process and Thread based
- d) None of the mentioned

Ans:c

5.What will happen if two thread of the same priority are called to be processed simultaneously? (CLO-4,L2)

- a) Anyone will be executed first lexicographically
- b) Both of them will be executed simultaneously
- c) None of them will be executed
- d) It is dependent on the operating system

Ans:d

6. Which of these statements is incorrect? (CLO-4,L2)

- a) By multithreading CPU idle time is minimized, and we can take maximum use of it
- b) By multitasking CPU idle time is minimized, and we can take maximum use of it
- c) Two threads in Java can have the same priority
- d) A thread can exist only in two states, running and blocked

Ans:d

7. Identify the technique that allows more than one program to be ready for execution and provides the ability to switch from one process to another. (CLO-4,L2)

- a) multitasking
- b) multiprocessing
- c) multitasking
- d) multiprogramming

Ans:d

8. The technique that increases the system's productivity. (CLO-4,L1)

- a) multiprogramming
- b) multitasking
- c) multiprocessing

d) single-programming

Ans:a

9. _____ is a property which more than one operation can be run simultaneously but it doesn't mean it will be. (CLO-4,L1)

- a. Concurrency
- b. Semaphore
- c. Mutual exclusion
- d. parallel process

Ans:a

10. _____ is a light-weight cooperatively-scheduled execution unit.(CLO-4,L3)

- a. gevent.Greenlet
- b. gevent.spawn()
- c. gevent.spawn_later()
- d. gevent.spawn_raw()

Ans:a

11. Which keyword is used to define methods in Python? (CLO-4,L2)

- (a) function
- (b) def
- (c) method
- (d) All of these

Ans:B

12. What is the correct translation of the following statement into mathematical logic? "Some real numbers are rational" .(CLO-4,L3)

- (A) $\exists x (\text{real}(x) \vee \text{rational}(x))$
- (B) $\forall x (\text{real}(x) \rightarrow \text{rational}(x))$
- (C) $\exists x (\text{real}(x) \wedge \text{rational}(x))$
- (D) $\exists x (\text{rational}(x) \rightarrow \text{real}(x))$

A

B

C

D

Ans: c

13. Which one of the following options is CORRECT given three positive integers x, y and z, and a predicate? .(CLO-4,L3)

$$P(x) = \neg(x=1) \wedge \forall y (\exists z (x=y*z) \Rightarrow (y=x) \vee (y=1))$$

$P(x)$ being true means that x is a prime number

$P(x)$ being true means that x is a number other than 1

$P(x)$ is always true irrespective of the value of x

$P(x)$ being true means that x has exactly two factors other than 1 and x

Ans: a

14. Suppose the predicate $F(x, y, t)$ is used to represent the statement that person x can fool person y at time t. which one of the statements below expresses best the meaning of the formula $\forall x \exists y \exists t (\neg F(x, y, t))$? .(CLO-4,L3)

- (a) Everyone can fool some person at some time
- (b) No one can fool everyone all the time
- (c) Everyone cannot fool some person all the time
- (d) No one can fool some person at some time

Ans: b

15. Which one of the following is the most appropriate logical formula to represent the statement? "Gold and silver ornaments are precious". (CLO-4,L3)

The following notations are used:

$G(x)$: x is a gold ornament

$S(x)$: x is a silver ornament

$P(x)$: x is precious

- (a) $\forall x(P(x) \rightarrow (G(x) \wedge S(x)))$
- (b) $\forall x((G(x) \wedge S(x)) \rightarrow P(x))$
- (c) $\exists x((G(x) \wedge S(x)) \rightarrow P(x))$
- (d) $\forall x((G(x) \vee S(x)) \rightarrow P(x))$

Ans : d

16. Which one of the first order predicate calculus statements given below correctly express the following English statement? .(CLO-4,L3)

Tigers and lions attack if they are hungry or threatened.

- (A) $\forall x [(\text{tiger}(x) \wedge \text{lion}(x)) \rightarrow \{(\text{hungry}(x) \vee \text{threatened}(x)) \rightarrow \text{attacks}(x)\}]$
- (B) $\forall x [(\text{tiger}(x) \vee \text{lion}(x)) \rightarrow \{(\text{hungry}(x) \vee \text{threatened}(x)) \wedge \text{attacks}(x)\}]$
- (C) $\forall x [(\text{tiger}(x) \vee \text{lion}(x)) \rightarrow \{\text{attacks}(x) \rightarrow (\text{hungry}(x) \vee \text{threatened}(x))\}]$
- (D) $\forall x [(\text{tiger}(x) \vee \text{lion}(x)) \rightarrow \{(\text{hungry}(x) \vee \text{threatened}(x)) \rightarrow \text{attacks}(x)\}]$

Ans: d

17. What is the first order predicate calculus statement equivalent to the following? (CLO-4,L3)

Every teacher is liked by some student

- (A) $\forall(x) [\text{teacher}(x) \rightarrow \exists(y) [\text{student}(y) \rightarrow \text{likes}(y, x)]]$
- (B) $\forall(x) [\text{teacher}(x) \rightarrow \exists(y) [\text{student}(y) \wedge \text{likes}(y, x)]]$
- (C) $\exists(y) \forall(x) [\text{teacher}(x) \rightarrow [\text{student}(y) \wedge \text{likes}(y, x)]]$

(D) $\forall (x) [\text{teacher}(x) \wedge \exists (y) [\text{student}(y) \rightarrow \text{likes}(y, x)]]$

Ans: b

- 18.
- I. $\neg \forall x (P(x))$ II. $\neg \exists x (P(x))$
III. $\neg \exists x (\neg P(x))$ IV. $\exists x (\neg P(x))$

Which of the above two are equivalent? (CLO-4,L3)

- (A) I and III
(B) I and IV
(C) II and III
(D) II and IV

Ans: b

19. _____ is a builtin python module where all possible types are defined. .(CLO-4,L2)

- (a) overload
b)typing
c)function
d)literal

Ans: b

20. _____ type represents a specific value of the specific type. .(CLO-4,L1)

- a) overload
b) typing
c) literal
d) None of the above

Ans: c

21. _____ is required to define multiple function declarations with different input types and results. .(CLO-4,L1)

- a) overload
- b) typing
- c) literal
- d) None of the above

Ans: a

PART B:(4 MARKS)

1. State parallel programming paradigm. (CLO-4, L1)
2. Differentiate parallel programming with functional programming. (CLO-4, L2)
3. Explain about Multithreading. (CLO-4, L1)
4. Explain about Multiprocessing. (CLO-4, L1)
5. Relate Serial processing concepts in Python. (CLO-4, L3)
6. Differentiate Serial Processing and Parallel Processing. (CLO-4, L3)
7. Demonstrate Multiprocessing module in Python. (CLO-4, L3)
8. Describe about Process class. (CLO-4, L2)
9. Design a Pool class in Python.(CLO-4, L3)
10. State Concurrent programming paradigm. (CLO-4, L1)
11. Compare multiprocessing and multitasking. (CLO-4, L2)
- 12.What are dependent functions ?(CLO4-L1)
13. Define typing module in dependent type programming? (CLO-4, L2)
- 14.Define dependent functions? (CLO-4, L2)
- 15.What is predicate logic? (CLO-4, L2)
- 16.Different types of quantifiers. (CLO-4, L1)

17.Explain Universal Quantifier and Existential Quantifier (CLO-4, L1)

18.Write some examples of Universal Qunatifier. (CLO-4, L2)

19 Define overload and literal in dependent type programming (CLO-4, L2)

20. Write the syntax for Existential Quantifier. (CLO-4, L2)

PART :C(12 MARKS)

1. Write a python program to implement producer consumer problem. (CLO-4, L3)

2. Implement the concept “Pool class” by importing a package pool. (CLO-4, L3)

3. Write a python program to implement dinning philosopher problem. (CLO-4, L3)

4. Explain the differences between multithreading and multiprocessing with example? (CLO-4, L1)

5. Write a program to implement thread synchronization (CLO-4, L3)

6. Compare Concurrent programming paradigm and functional programming paradigm with example program. (CLO-4, L2)

7. Explain in detail about dependent type programming .(CLO4-L1)

8. Write a python program to create Type aliases using typing module.. (CLO-4, L3)

9. Write a python program to check every **key:value** pair in a dictionary and check if they match the **name@email** format using typing module.. (CLO-4, L3)

10. Write a python program to create new user defined type Student using typing module.. (CLO-4, L3)

11. Write a python program for function dependent type using overload and literals .. (CLO-4, L3)

1. What does a threading.Lock do?

- a) Pass messages between threads
- b) Allow only one thread at a time to access a resource**
- c) Wait until a thread is finished
- d) SHIFT TO ALL CAPS

2. What does the Thread.join() method do?

- a) Waits for the thread to finish**
- b) Restricts access to a resource
- c) Adds the thread to a pool
- d) Merges two threads into one

3) Race conditions are ...

- a) Testing which thread completes first
- b) Two threads incorrectly accessing a shared resource**
- c) The weather on race day
- d) Something you should add to your code

4) It sets the lock state to locked. If called on a locked object, it blocks until the resource is free.

- a) lock()
- b)release()
- c) acquire()**
- d)join()

5) You have thread T1, T2, and T3. How will you ensure that thread T2 is run after T1 and thread T3 after T2?

- a) Sleep method
- b) Join Method**
- c)Release
- d)lock method

6) What are the libraries in Python that support threads?

- _threading
- b) Thread
- c)None
- d)thread**

7) Mention the correct syntax for creating Thread object for calling increment methods

- a) t1 = threading.Thread()
- b) t1 = threading.Thread(target)
- c) t1 = threading.Thread(target=incr)**
- d) t1 = threading.Thr

8) Which leads to concurrency?

- a) Serialization
- b) Parallelism**
- c) Serial processing
- d) Distribution

9) Which is not a method for parallelism?

- a) Message Passing
- b) Shared Memory
- c) Threads
- d) Sockets**

10) A race condition occurs when multiple processes or threads read and write

- a). Input b). Information c). **Data Items** d). Programs

11) For a single processor system, implementation of semaphores is possible to inhibited through

- a) Deadlock b) **Interrupts** c) Lock Step d) Paging

12) Which method controls the execution of thread in python?

- a) Wait b) **Sleep** c) Acquire d) Lock

13) How does run() method is invoked?

- a) By Thread.create() b) **By Thread.start()**

- c) None d) By Thread.run()

14) What is the difference between *threading.Lock* and *threading.RLock*?

- a) Lock and RLock both primitives are owned by a single thread.

- b) **Lock is owned by none while RLock is owned by many.**

- c) Lock is owned by a thread while RLock is owned by many.

- d) Lock and RLock both primitives are owned by many.

15) What is the difference between a *semaphore* and *bounded semaphore*?

- a) Semaphore holds a counter for the number of release() calls minus the number of acquire() calls, plus an initial value but bounded semaphore doesn't.

- b) A semaphore makes sure its current value doesn't exceed its initial value while bounded semaphore doesn't.

- c) **A bounded semaphore makes sure its current value doesn't exceed its initial value while semaphore doesn't.**

- d) Bounded semaphore holds a counter for the number of release() calls minus the number of acquire() calls, plus an initial value but semaphore doesn't.

16) What is the method that wakes up all thread waiting for the condition?

- a) releaseAll() b) notify() c) **notifyAll()** d) release()

17 How to terminate a blocking thread?

- a). **thread.stop() & thread.wait()** b) thread.stop()

- c) thread.terminate() d) None

18. Which synchronization method is used to guard the resources with limited capacity, e.g. a database server?

- a) Event b) Condition c) Lock d) **Semaphore**

19. How to detect the status of a python thread?

- a) isActive()
- b) isDaemon()
- c) None
- d) **isAlive()**

20. Execution of several activities at the same time.

- a) processing
- b) parallel processing**
- c) serial processing
- d) multitasking

1.Sympy is a collection of mathematical algorithms and convenience functions built on the -----extension of Python

- a) numpy
- b) scikit
- c) sys
- d) functools

2. Exponential function computes the -----

- a) 10^{**x} element-wise
- b) 10^{**x} row-wise
- c) 10^{**x} column-wise
- d) 10^x element-wise

3. _____ evaluates the expression to a floating-point number.

- a) evalf
- b) fval
- c) float
- d) valf

4. what is the output for the following expression $((x+y)^{**2}).expand()$

- a) $x^2 + 2xy + y^2$
- b) $x^2 + 2*x*y + y^2$
- c) $x^2 + 2*x*y + y^2$
- d) $x^{**2} + 2*x*y + y^{**2}$

5. limit($(5^{**x} + 3^{**x})^{**}(1/x)$, x, oo) ,what is the output

- a) 5
- b) oo
- c) 1
- d) 0

6) Higher derivatives can be calculated using the which method

- a) higher(func,var,n)
- b) diff(func, var, n)
- c) diff(n,var,func)
- d) diff(func, var)

7) what is the output

```
>>> x = Symbol('x')  
>>> y = Symbol('y')  
>>> A = Matrix([[1,x], [y,1]])  
>>> A**2
```

- a) $[1, x]$
 $[y, 1]$
- b) $[xy + 1, 2x]$
 $[2y, xy + 1]$
- c) $[x*y + 1, 2*y]$
- d) $[x*y + 1, 2*x]$

[$2*x$, $x*y + 1$]

[$2*y$, $x*y + 1$]

8) .match() method, along with the ----- class, to perform pattern matching on expressions.

- a) pattern
- b) func
- c) wild**
- d) dictionary

9) which among the following function Return or print, respectively, a pretty representation of expr

- a) pretty(expr)
- b) pretty_print(expr)
- c) pprint(expr)
- d) all of the above**

10) What is the output of

```
from sympy.abc import a, b
```

```
expr = b*a + -4*a + b + a*b + 4*a + (a + b)*3
```

- a) $ba-4a+b+ab+4a+3(a+b)$
- b) $2*a*b + 3*a + 4*b$**
- c) $2ab+3a+4b$
- d) all of the above

11) print(pi.evalf(30))

- a) $3.14/30$
- b) $30/3.14$
- c) 3.14159265358979323846264338328**
- d) 3.14

12) which is the correct way to write equation for $x^2=x$ in sympy

- a) $x^{**2} = x$**
- b) $x*x = x$
- c) $x\%2 = x$
- d) $X^2 = x$

13) how to find a solution for a equation in a given interval

- a) solve(equation,range)
- b) equation(solve,range)
- c) solveset()**
- d) both a and b

14) Allows , the same elements can appear multiple times at different positions

a) set **b)sequence**

c) dictionary d) none

15) which is the snippet to find the eigenvalues of $\begin{bmatrix} 1 & 2 \\ 2 & 2 \end{bmatrix}$

a) `Matrix([[1, 2], [2, 2]]).eigenvals()` b) `Matrix([[1, 2], [2, 2]]).eigen`

c) both a and b d) `eigen([[1, 2], [2, 2]])`

16 What is the purpose of sympify() method?

- a. Convert expression of string type to mathematical expression**
- b. Convert mathematical expression to String
- c. Convert mathematical expression to character
- d. Convert tuple to mathematical expression

17 Find the output of the following program

```
from sympy import solve
x = Symbol('x')
expr = x**2 + 5*x + 4
solve(expr, dict=True)
```

- a. [{x: -4}, {x: -1}]**
- b. [{x: -6}, {x: -1}]
- c. [{x: -1}, {x: -4}]
- d. [{x: 4}, {x: -1}]

18. A *rational expression* is an algebraic expression in which the numerator and denominator are both -----

- a. Equal
- b. Polynomials**
- c. Unequal
- d. Symmetric

19. Find the output of the following program

```
# import sympy
from sympy import *
```

```

x = symbols('x')
expr = sin(x)/x;
print("Expression : {}".format(expr))

# Use sympy.limit() method
limit_expr = limit(expr, x, 0)
print("Limit of the expression tends to 0 : {}".format(limit_expr))

```

- a. Expression : $\cos(x)/x$
Limit of the expression tends to 0 : 2
- b. Expression : $\tan(x)/x$
Limit of the expression tends to 0 : 3
- c. Expression : $\sin(x)/x$
Limit of the expression tends to 1 : 0
- d. **Expression : $\sin(x)/x$**
Limit of the expression tends to 0 : 1

20. ----- method will simplify mathematical expression using trigonometric identities.

- a. **sympy.trigsimp()**
- b. sympy.series()
- c. sympy.lambda()
- d. sympy.sim()

Which of the following programming paradigms allow us to write programs and know they are correct before running them?

- a) Automata based Programming Paradigm
- b) Logical Programming Paradigm
- c) **Dependent type Programming Paradigm**
- d) Imperative Programming Paradigm

2) Which of the following is false regarding dependent types?

- a) They allow us to write programs and know they are correct before running them.
- b) **They allow us to write programs and know they are correct only after running them.**
- c) You can specify types that can check the value of your variables at compile time.
- d) Its definition depends on a value.

3) Which of the notations is true for “P(x) is true for all values of x in the universe of discourse”?

- a) $x\forall P(x)$
- b) $\forall P(x)x$
- c) $\exists xP(x)$
- d) **$\forall xP(x)$**

4) Which of the following is false about quantifiers?

- a) Notation \forall is used for Universal quantifier.
- b) Notation \exists is used for Existential quantifier.
- c) “All men drink tea” is an example of Universal Quantifier
- d) **“All men drink coffee” is an example of Existential Quantifier.**

5) Let P(x) be the predicate “x must take a discrete mathematics course” and let Q(x) be the predicate “x is a computer science student”.

Which of the following statements is correct for “Everybody must take a discrete mathematics course or be a computer science student”?

- a) **$\forall x(Q(x) \vee P(x))$**
- b) $\forall x(Q(x)) \vee \forall x(P(x))$
- c) $\forall x(Q(x) \parallel P(x))$
- d) $\forall x(Q(x) \rightarrow P(x))$

6) Which of the following is correct for predicate “All men drink coffee”?

- a) $\forall x \text{men}(x) \rightarrow (x, \text{coffee})$
- b) $\text{drink}(x, \text{coffee}) \rightarrow \forall x \text{men}(x)$

- c) $\forall x \text{ men}(x) \rightarrow \text{drink}(x, \text{coffee})$
d) $\forall \text{men}(x) \rightarrow \text{drink}(x, \text{coffee})$
- 7) Which of the following is correct for predicate "Some boys play football"?
a) $\exists x \text{ boys}(x) \rightarrow \text{play}(x, \text{football})$
b) $\exists \text{ boys}(x) \rightarrow \text{play}(x, \text{football})$
c) $\forall x \text{ boys}(x) \rightarrow \text{play}(x, \text{football})$
d) $\exists x \forall \text{boys}(x) \rightarrow \text{play}(x, \text{football})$
- 8) Dependent Type is used to encode _____ like "for all" and "there exists".
a. **Logic Quantifiers**
b. Analysing Quantifiers
c. Dependent Quantifiers
d. None of the above
- 9) "There exists x in the universe of discourse such that P(X) is true" is which Quantifiers statement?
a. Logical
b. Universal
c. **Existential**
d. None of these
- 10) What is the way to determine if a given function is dependant type
a. Result is independent of the argument
b. Result depends upon the usage in the program
c. **Result depends on the Value of its argument**
d. Result depends on available resources
- 11) Notation for an Existential Quantifier:
a. $\forall x P(x)$
b. $\Sigma P(x)$
c. $\emptyset x P(x)$
d. **$\exists x P(x)$**
- 12) Representation of the following statement:
Every Clock has quartz
a. $\emptyset x \text{clock}(x) \rightarrow \text{quartz}(x)$
b. $\exists x \text{clock}(x) \rightarrow \text{quartz}(x)$
c. **$\forall x \text{clock}(x) \rightarrow \text{quartz}(x)$**
d. none of the above
- 13) Representation of the following statement:
Some leaves are Red

a. $\exists x \text{leaves}(x) \rightarrow \text{red}(x)$

b. $\forall x \text{leaves}(x) \rightarrow \text{red}(x)$

c. $\Sigma \text{leaves}(x) \rightarrow \text{red}(x)$

d. none of the above

13. A function has dependent type if the _____ of a function's result depends on the _____ of its Argument

a. value and type

b. type and value

c. type and type

d. value and value

14. Dependent type paradigm used to encode logic's quantifiers like _____ and _____

a. for one, there may exists

b. for all, there always

c. for all, there exists

d. for specific, there exists

15. Choose the correct one with respect to typing and typing-extensions library in python dependent type programming.

a. typing is not builtin python module where all possible types are defined.

typing_extensions is an official package for new types in the future releases of python

b. typing is a builtin python module where all possible types are defined.

typing_extensions is an official package for new types in the future releases of python

c. typing is a builtin python module where all possible types are defined.

typing_extensions is not an official package for new types in the future releases of python

d. typing is not a builtin python module where all possible types are defined.

typing_extensions is not an official package for new types in the future releases of python

16. Predict the output of the below mentioned python code without dependent type syntax:

```
def make_hamburger(meat, number_of_meats):
    return ["bread"] + [meat] * number_of_meats + ["bread"]
print(make_hamburger("ground beef", 2))
```

a. ['bread', 'ground beef', 2, 'bread']

b. ['bread', 'ground beef', 'ground beef', 'bread']

c. *TypeError: cannot concatenate 'str' and 'int' objects*

d. `['bread', 'ground beef', '2bread']`

17. Predict the output of the below mentioned python code with dependent type syntax:

```
from typing import List

def greet_all(names: List[str]) -> None:
    for name in names:
        print('Hello ' + name)

names = ["Alice", "Bob", "Charlie"]
ages = [10, 20, 30]

greet_all(names)
greet_all(ages)

a. greet_all(names) # Ok!
    greet_all(ages) # Ok!

b. a.greet_all(names) # Ok!
    greet_all(ages) # Error due to incompatible types

c. a.greet_all(names) # Error due to incompatible types
    greet_all(ages) # Ok!

d. a.greet_all(names) # Error due to incompatible types
    greet_all(ages) # Error due to incompatible types
```

18 Let x be a variable which refers to Universe of Disclosure such as x1,x2....xn then

,how to represent this statement using quantifiers “ All Man working in Industry”

- a) $\forall x \text{ man}(x) \rightarrow \text{work}(x, \text{Industry})$ b) $\forall x \text{ man}(x) \rightarrow \text{work}(\text{industry})$.
- a) $\forall x \rightarrow \text{work}(x, \text{industry})$ d) $\forall x \text{ man}(x) \rightarrow \text{Industry}(\text{work})$

19 How do you represent the statement “Every man respects his parent.”

- a) $\forall x \text{ woman}(x) \rightarrow \text{respects}(x, \text{parent})$ b) $\forall x \text{ man}(x) \rightarrow \text{respects}(x, \text{parent})$
- c) $\forall x \text{ man} \rightarrow \text{respects}(x, \text{parent})$ d) $\forall x \text{ man}(x) \rightarrow \text{respects}(\text{parent})$.

20 How do you represent the statement “Some boys play cricket “

- a) $\exists x \text{ boys}(x) \rightarrow \text{play}(\text{all})$ b) $\forall x \text{ boys}(x) \rightarrow \text{play}(x, \text{cricket})$
- c) $\exists x \text{ boys}(x) \rightarrow \text{play}(x, \text{cricket})$ d) $\exists x \wedge \forall x \text{ boys}(x) \rightarrow \text{play}(x, \text{cricket})$

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
VADAPALANI CAMPUS
DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

18CSC207J – ADVANCED PROGRAMMING PRACTICES

NETWORK PROGRAMMING PARADIGM

1. _____ programming is a way of connecting two nodes on a network to communicate with each other.
 - i. Logic
 - ii. **Socket**
 - iii. Functional
 - iv. Symbolic

2. Which method used by the server to initiate the connection with the client?
 - i. Listen()
 - ii. Close()
 - iii. Bind()
 - iv. **Accept()**

3. _____ is a socket through which data can be transmitted continuously.
 - i. Datagram Socket
 - ii. **Stream Socket**
 - iii. Raw Socket
 - iv. Binary Socket

4. _____ paradigm deals with client – server communication
 - i. Dependent type paradigm
 - ii. Parallel programming paradigm
 - iii. Network paradigm
 - iv. Concurrent programming paradigm

5. _____ protocol facilitates sending of datagrams in an unreliable manner.

- i. TCP
 - ii. UDP**
 - iii. HTTP
 - iv. FTP
- 6. Which among methods are not server socket?
 - i. connect()**
 - ii. bind()
 - iii. listen()
 - iv. accept()
- 7. In UDP, Which among methods are used to receive messages at endpoint
 - i. sock_object.recv()
 - ii. sock_object.send()
 - iii. sock_object.recvfrom()**
 - iv. sock_object.sendto()
- 8. In TCP, Which among methods are used to send messages from endpoint
 - i. sock_object.recv()
 - ii. sock_object.send()**
 - iii. sock_object.recvfrom()
 - iv. sock_object.sendto()
- 9. The protocol which defines IPv4 is
 - i. AF_UNIX
 - ii. SOCK_STREAM
 - iii. AF_INET**
 - iv. SOCK_DGRAM
- 10. The correct order of methods used in server socket is
 - i. Socket(), Bind(), listen(), accept()**
 - ii. Socket(), listen(), bind(), accept()
 - iii. Socket(), accept(), bind(), listen()
 - iv. Socket(), bind(), accept(), listen()

11. _____ is a type of network socket which provides connection less point for sending and receiving packets.

- i. **Datagram Socket**
- ii. Stream Socket
- iii. Raw Socket
- iv. Binary Socket

12. _____ do not use any transport protocol but data is directly transmitted over IP protocol

- i. Datagram Socket
- ii. Stream Socket
- iii. **Raw Socket**
- iv. Binary Socket

13. The _____ is a physical path over which the message travels.

- i. Protocol
- ii. **Medium**
- iii. Path
- iv. Route

14. A pair (host, port) is used for the _____ address family.

- i. AF_NETLINK
- ii. AF_INET6
- iii. **AF_INET**
- iv. AF_ALG

15. Use _____ to make the socket to visible to the outside world.

- i. socket.listen()
- ii. socket.visible()
- iii. socket.socket()
- iv. **Socket.gethostname()**

16. In which mode socket is created in default.

- i. **Blocking mode**
- ii. Non-Blocking Mode
- iii. Timeout mode
- iv. Accept mode

17. Which method is recommended to be called before calling connect() method?

- i. getdefaulttimeout()

- ii. **settimeout()**
- iii. **getaddrinfo()**
- iv. no such method

18. Which exception is raised for address related errors by getaddrinfo()?

- i. gaoerror
- ii. gsierror
- iii. **gaierror**
- iv. gdeerror

19. _____ protocol facilitates sending of datagrams in an reliable manner.

- i. **TCP**
- ii. UDP
- iii. HTTP
- iv. FTP

20. To create a socket, which function among the following is available in python socket module?

- i. socket.create()
- ii. socket.initialize()
- iii. **socket.socket()**
- iv. socket.build()

UNIT V

PART A: MULTIPLE CHOICE QUESTIONS

Symbolic:

1. Which of the following is false about sympy? (CLO5-L1)
 - a. Sympy is a python library for symbolic mathematics
 - b. It requires external libraries for execution
 - c. It is an alternative to the systems like mathematica or maple

Ans: B

2. Symbols can now be manipulated using some of python operators using _____.(CLO5-L1)
A)+. B) && C) ? D\$

Ans: A

3. Which of the following belongs to the numerical type of sympy (CLO5-L1)
 - a. Float
 - b. Integer
 - c. Decimal
 - d. Factorial

Ans: B

4. Choose the correct output for the following code?(CLO5-L2)

```
Import sympy as sym
```

```
a= sym.Rational(4,6)
```

```
print a
```

a. 6/4

b. 0.66

c. 4/6

d. 1.5

Ans: C

5. What is the output for the trigonometry function? (CLO5-L2)

```
Sym.expand(sym.cos(x+y),trig=true)
```

a. sin(x)*sin(y)+cos(x)*cos(y)

b. sin(x)*cos(y)+cos(x)*sin(y)

c. -sin(x)*sin(y) - cos(x)*cos(y)

d. -sin(x)*sin(y) + cos(x)*cos(y)

Ans: D

6. Differentiate the Sympy Expression using the syntax (CLO5-L1)

a. diff (var,func)

b. diff(func,var)

c. diff(expr,var)

d. diff(var,point)

Ans: B

7. SymPy is able to solve algebraic equations, in one and several variables using(CLO5-L1)

- a. solveset()
- b. series()
- c. real()
- d. limit()

Ans: A

8. Which of the following belongs to the calculus function? (CLO5-L1)

- i. Limit
 - ii. Series
 - iii. Arithmetic
 - iv. Computation
- a. Both i,ii
 - b. Both ii,iii
 - c. Both ii,iv
 - d. Both i,iii,iv

Ans: A

9. Choose the output for the following code? (CLO5-L2)

Limit (sin(x), x,0)

- a. 0
- b. 1
- c. Infinite
- d. Error

Ans: B

10. Which of the following is the correct output for the below given code? (CLO5-L2)

Sym.integrate(x**3, (x,-1,1)

- a. 1
- b. 3
- c. 0
- d. -1

Ans: C

11. Which of the following is the correct output for the below given code? (CLO5-L2)

x,y=sym.symbols('x,y')

A=sym.Matrix([[1,x],[y,1]])

Print A

- a. Matrix ([[1,x], [y,1]])
- b. Matrix ([[x,1],[1,y]])
- c. Matrix ([[0,x], [y,0]])
- d. Matrix ([[x,0],[0,y]])

Ans: A

12. Choose the output for the following code? (CLO5-L2)

Sym.solveset(x**4-1,x)

- a. {1,-1,I,-I}
- b. {-1,1,-I,I}
- c. {0,1,I,-I}
- d. {1,-1,-I,I}

Ans: B

13. Limit the Sympy Expression using the syntax (CLO5-L1)
- a. limit (var,func,point)
 - b. limit(func,var,point)
 - c. limit(func,var)
 - d. limit(var,point)

Ans: B

14. Finite state machines are used for____(CLO5-L1)

- a. Pseudo random test patterns
- b. Deterministic test patterns
- c. Random test patterns
- d. Algorithmic test patterns

Ans:D

15. According to the given transitions, which among the following are the epsilon closures of q1 for the given NFA? (CLO5-L3)

$$\Delta(q_1, \varepsilon) = \{q_2, q_3, q_4\}$$

$$\Delta(q_4, 1) = q_1$$

$$\Delta(q_1, \varepsilon) = q_1$$

- a. q4
- b. q2
- c. q1
- d. q1, q2, q3, q4

Ans: D

16. Which of the following tuples order are correct for Deterministic Finite Automata (DFA) (CLO5-L1)

- a. (Q, S, d, q_0, F)
- b. (Q, S, d, F, q_0)
- c. (S, Q, d, q_0, F)
- d. (Q, S, q_0, d, F)

Ans: A

17. NFA, is called as 'non-deterministic' because of :(CLO5-L1)

- a. The result is undetermined
- b. The choice of path is non-deterministic
- c. The state to be transited next is non-deterministic
- d. All of the mentioned

Ans: B

18. _____ is a class attribute defined by its source state and destination state. .(CLO5-L1)

- a. LGPL
- b. Scipy
- c. Transition
- d. State

Ans : C

19. Among the following which is used to define the behavior of what we want to achieve (CLO5-L1)

- a. State
- b. Automata
- c. Transition
- d. Machine

Ans: D

20. _____is a triplet consisting of two states and one command needed for the change of one state to the other. .(CLO5-L1)

- a. machine
- b. Automata
- c. State
- d. Transitions

Ans:D

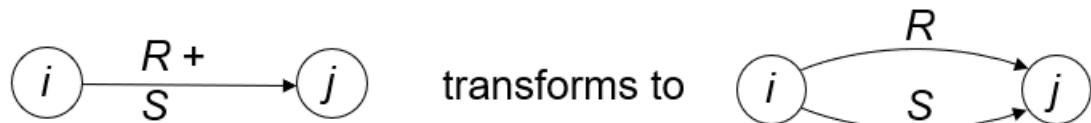
21. _____is a mathematical model of computation. .(CLO5-L1)

- a. state machine
- b. Automata
- c. State
- d. Transitions

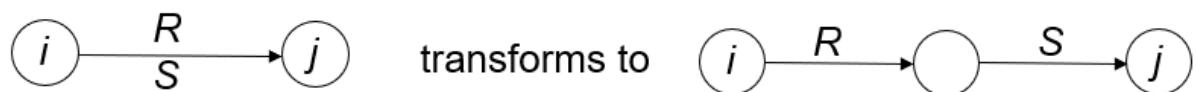
Ans: A

22. Which of the following is correct among the following expressions? (CLO5-L2)

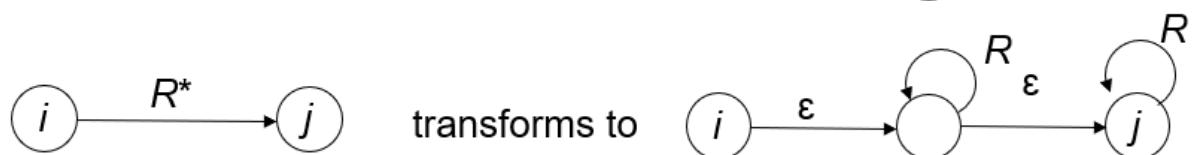
i.



ii.



iii.



- a. Both i,ii
- b. Both ii,iii
- c. Both i,iii
- d. All of the above

Ans: A

23. What kind of abstract machine can recognize strings in a regular set? (CLO5-L1)

- a. DFA
- b. NFA
- c. PDA
- d. None of the above

Ans: A

24. Which of the following statements is wrong? (CLO5-L1)

- a. The language accepted by finite automata are the languages denoted by regular expressions
- b. For every DFA there is a regular expression denoting its language
- c. For a regular expression r , there does not exist NFA with $L(r)$ any transit that accept
- d. None of the above.

Ans: C

25. Let for $\Sigma = \{0,1\}$ $R = (\Sigma\Sigma\Sigma)^*$, the language of R would be-----
(CLO5-L2)

- a. $\{w \mid w \text{ is a string of odd length}$
- b. $\{w \mid w \text{ is a string of length multiple of 3}\}$
- c. $\{w \mid w \text{ is a string of length 3}\}$
- d. All of the above.

Ans: B

26. $(a+b)^*$ is equivalent to----- (CLO5-L2)

- a. b^*a^*
- b. $(a^*b^*)^*$
- c. a^*b^*
- d. None of the above.

Ans: B

27. In regular expressions, the operator '*' stands for----- (CLO5-L1)

- a. Concatenation
- b. Addition
- c. Selection
- d. Iteration

Ans: D

28. The regular expression with all strings of 0's and 1's with at least two consecutive 0's is----- (CLO5-L2)

- a. $1 + (10)^*$
- b. $(0+1)^*011$
- c. $(0+1)^*00(0+1)^*$
- d. $0^*1^*2^*$

yh

Ans: C

29. Which of the following operation can be applied on regular expressions? (CLO5-L1)

- a. Union
- b. Concatenation
- c. Closure
- d. All of the above

Ans: D

30. Finite state machine will initially set to all zeroes. (CLO5-L1)

- a. True
- b. False

Ans: A

31. _____ allow millions of different machines, using all sorts of different network hardware, to pass packets to each other over the fabric of an IP network.(CLO5-L1)

- a)Sockets

- b)Client machine
- c)server machines
- d)IP address

Ans: d

32. Which are all necessary to direct a packet to its destination.(CLO5-L2)

- a)IP Address
- b)IP address and port
- c)port
- d)InternetAssignedNumbers

Ans:b

33. _____ (0–1023) are for the most important and widely-used protocols(CLO5-L1)

- a. Well Known ports
- b)Registered ports
- c)Known ports
- d)Unregistered ports

Ans:a

34. _____ (1024–49151) are not usually treated as special by operating systems(CLO5-L1)

- a)Registered ports
- b)well known ports
- c)Known ports
- d)Unregistered ports

Ans:a

35. Python's standard socket module supports _____(CLO5-L1)

- a)getByName()
- b)getName()
- c)gethostbyname()
- d)getServName()

Ans:c

36. Example for Connection oriented protocol(CLO5-L1)

- a)UDP
- b)SMTP
- c)FTP
- d)TCP

Ans:d

37. Example for Connection less protocol(CLO5-L1)

- a)SMTP
- b)FTP
- c)UDP
- d)TCP

Ans:c

38. _____ is an application-level block of transmitted data.(CLO5-L1)

- a)data
- b)datagram
- c)segmentation
- d)Fragmentation

Ans:b

39. Simple server uses _____ command to request a UDP network address.(CLO5-L1)

- a)bind()
- b)socket()
- c)getName()
- d)getsockName()

Ans:a

40. Which method is used by python program to retrieve the current IP and port to which the socket is bound. (CLO5-L2)

- a)bind()
- b)getsockname()
- c)getName()
- d)getSocket()

Ans:b

41. Sending packets with another computer's return address is called _____(CLO5-L1)

- a)Binding
- b) Unification
- c)Spoofing
- d) IP address

Ans:c

42. The concept by which larger UDP packets are spitted into several small physical packets(CLO5-L1)

- a)fragmentation
- b)Binding
- c) Unification
- d) partition

Ans:a

43. The MTU is _____ that all of the network devices between two hosts will support. (CLO5-L1)

- a)smallest packet size
- b)medium packet size
- c)average packet size
- d)largest packet size

Ans:d

44. _____ is efficient only if your host ever only sends one message at a time, then waits for a response.(CLO5-L1)

- a)UDP
- b)TCP
- c)FTP
- d)SMTP

Ans: a

45. Which protocol provides reliable connection?(CLO5-L2)

- a)UDP
- b) FTP
- c)TCP
- d)SMTP

Ans:c

46. TCP uses a _____ that counts the number of bytes transmitted.(CLO5-L1)

- a) protocol
- b)counter
- c)rules
- d)Ports

Ans:b

47. The amount of data that a sender is willing to have on the wire at any given moment is called _____(CLO5-L1)
a) The size of the TCP window. b) counter c) port d) protocol

Ans:a

48. TCP uses _____ to distinguish different applications running at the same IP address, and follows exactly the same conventions regarding well-known and ephemeral port numbers.(CLO5-L1)
a) IP address b)port c)port numbers d) socket ID
Ans:c

49. _____refers to the address family ipv4.(CLO5-L1 (CLO5-L1))
a) Af_INET b) AS_INET c) AG_INET d) AN_INET

Ans:a

50. A server has a _____method which puts the server into listen mode.(CLO5-L1)
a)list() b)listento() c)listen() d)listenfrom()

Ans:c

51. _____ are the end-point of a two-way communication .(CLO5 L1) a)IP Address b)sockets c)sockets ID d) Object ID
Ans:b

52. Identify the method used to connect the client to host and port and initiate the connection towards the server.(CLO5-L3)
a)sock_object.recv() b)sock_object.send()
c)sock_object.append() d)sock_object.connect():
Ans:d

53. Identify the method to receive messages at endpoints when the value of the protocol parameter is TCP.(CLO5-L3)
a)sock_object.recv() b)sock_object.send()
c)sock_object.append() d)sock_object.connect():
Ans:a

54. Which method is used to receive messages at endpoints if the protocol used is UDP.(CLO5-L2)
a)sock_object.recv() b)sock_object.recvfrom()
c)sock_object.append() d)sock_object.connect():
Ans:b

55. Identify the method that returns host name.(CLO5-L1)

- a)sock_object.recv() b)sock_object.recvfrom()
c)sock_object.gethostname d)sock_object.connect():
Ans:c

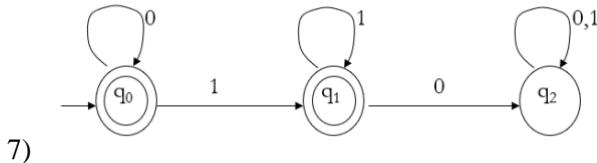
56. Identify the method to send messages from endpoints if the protocol parameter is UDP.(CLO5-L3)
a)sock_object.sendto() b)sock_object.recvfrom()
c)sock_object.gethostname d)sock_object.connect():
Ans:a

57. _____can be used to end direction of communication in a socket(CLO5-L1)
a. The shutdown() call b)Shut() c)close() d)end()

Ans:c

PART: B (4MARKS)

- 1) Write a program to factorize the following expression. (CLO5-L3)
 $x^{**3} + 3*x^{**2}*y + 3*x*y^{**2} + y^{**3}$
- 2) What is SymPy? (CLO5-L1)
- 3) Write the commands to perform the operations on substitutions and expressions (CLO5-L1)
- 4) Design a DFA that accepts all strings that contain 010 or do not contain 0. (CLO5-L2)
- 5) Differentiate Finite state machine and Non deterministic Finite state machine. (CLO5-L2)
- 6) Let $\Sigma = \{0, 1\}$. Give DFAs for $\{\}$, $\{\epsilon\}$, Σ^* , and Σ^+ . (CLO5-L2)



- 7) Provide the DFA function for the above-mentioned diagram (CLO5-L2)

- 8) Compare the features of LISP and Wolfram. (CLO5-L2)
- 9) Write a DFA automata code for $L(M) = \{ w \mid w \text{ has an even number of } 1s \}$ (CLO5-L2)
- 10) Write a DFA automata code for $L(M) = \{(ab)^n \mid n \in \mathbb{N}\}$ (CLO5-L2)
- 11) Construct NFA that recognizes the language of strings that end in 01 (CLO5-L2)
- 12) Find an NFA to recognize the language $(a + ba)^*bb(a + ab)^*$. (CLO5-L2)
- 13) Write the features of GUI programming. (CLO5-L1)
- 14) Explain briefly Automata based programming in python. (CLO5-L1)
- 15) Write the syntax for Expand and Factor command. Give example. (CLO5-L1)
- 16) Give example for DFA and explain. (CLO5-L2)
- 17) Differentiate Python and Jython. (CLO5-L2)
- 18) Write the syntax for series and Integration command. Give example .(CLO5-L1)
- 19) Give example for NFA and explain. (CLO5-L1)
- 20) Differentiate TCP and UDP(CLO5-L2)
- 21) Compare connection oriented and connectionless service(CLO5-L2)
- 22) Differentiate automatic and manual configuration(CLO5-L2)
- 23) How will you generate random port numbers?(CLO5-L2)
- 24) What is file descriptors? Give example(CLO5-L1)
- 25) Differentiate bind() and getSockName() .(CLO5-L2)
- 26) What is UDP fragmentation? Give example .(CLO5-L1)
- 27) What will happen when you send large packets? Give example.(CLO5-L2)
- 28) Differentiate multicast and broadcast.(CLO5-L2)
- 29) Write the code to connect server and client in TCP.(CLO5-L3)
- 30) How does TCP provides reliable connection.(CLO5-L2)

- 31) Compare passive and active socket.(CLO5-L2)
 32) Write the code to connect server and client in UDP.(CLO5-L3)

PART-C (12 MARKS)

1. Write a program to expand and factorize the following expression. (CLO5-L3)

- a. $x^3 + 3x^2y + 3xy^2 + y^3 = (x + y)^3$
- b. $x + y + x^*y$

2. Consider the following series:

$$X + (X^2/2) + (X^3/3) + (X^4/4) + \dots + (X^n/n)$$

Write a python program that will ask a user to input a number, n, and print this series for that number. In the series, x is a symbol and n are an integer input by the program's user. The nth term in this series is given as (X^n/n) . (CLO5-L3)

- 3. Write a program to implement client server communication using UDP. (CLO5-L3)
 - 4. Write a program to demonstrate the concept of UDP fragmentation and explain .(CLO5-L3)
 - 5. Explain automata-based programming paradigm. (CLO5-L1)
 - 6. Design go, slowdown and stop events according to the traffic scenario to cross the road using python code and also draw transition diagram and transition table for the same scenario. (CLO5-L3)
 - 7. Write a example program to find limits, differentiation, Series and Integration. (CLO5-L1)
 - 8. Differentiate DFA and NFA. Explain NFA with example. (CLO5-L2)
 - 9. Build an automaton that accepts all and only those strings that contain 001 (CLO5-L2)
 - 10. Find an DFA for each of the following languages over the alphabet {a, b}.(CLO5-L3)
 - (a) $\{(ab)^n \mid n \in \mathbb{N}\}$, which has regular expression $(ab)^*$.
 - b) Find a DFA for the language of $a + aa^*b$.
 - 11.a. Write NFA automata code for the Language that accepts all end with 01 (CLO5-L3)
 - b. Write a automata code for $L(M) = a + aa^*b + a^*b$.
 - c Write a automata code for Let $\Sigma = \{0, 1\}$.
- Given NFAs for $\{\}$, $\{\epsilon\}$, $\{(ab)^n \mid n \in \mathbb{N}\}$, which has regular expression $(ab)^*$.
12. Write a program to convert NFA to DFA (CLO5-L2)
13. Explain in detail about communication using UDP with example. .(CLO5-L1)

UNIT 4

Dependent Programming paradigm, parallel and concurrent Programming paradigm

1. Parallelism representation is critical to the success of ----- (CLO-4,L1)

- a)High-performance computing.
- b)Low-performance computing
- c)Scaling
- d)Vectorization

Ans:a

2. Parallel programming through a combination of -----and ----- (CLO-4,L1)

- a.Patterns, examples
- b.Algorithms , flowcharts
- c.Models , methods
- d.Classes ,objects

Ans:a

3.What is multithreaded programming? (CLO-4,L1)

- a) It's a process in which two different processes run simultaneously
- b) It's a process in which two or more parts of same process run simultaneously
- c) It's a process in which many different process are able to access same information
- d) It's a process in which a single process can access information from many sources

Ans:b

4. Which of these are types of multitasking? (CLO-4,L2)

- a) Process based
- b) Thread based
- c) Process and Thread based
- d) None of the mentioned

Ans:c

5.What will happen if two thread of the same priority are called to be processed simultaneously? (CLO-4,L2)

- a) Anyone will be executed first lexicographically
- b) Both of them will be executed simultaneously
- c) None of them will be executed
- d) It is dependent on the operating system

Ans:d

6. Which of these statements is incorrect? (CLO-4,L2)

- a) By multithreading CPU idle time is minimized, and we can take maximum use of it
- b) By multitasking CPU idle time is minimized, and we can take maximum use of it
- c) Two threads in Java can have the same priority
- d) A thread can exist only in two states, running and blocked

Ans:d

7. Identify the technique that allows more than one program to be ready for execution and provides the ability to switch from one process to another. (CLO-4,L2)

- a) multitasking
- b) multiprocessing
- c) multitasking
- d) multiprogramming

Ans:d

8. The technique that increases the system's productivity. (CLO-4,L1)

- a) multiprogramming
- b) multitasking
- c) multiprocessing

d) single-programming

Ans:a

9. _____ is a property which more than one operation can be run simultaneously but it doesn't mean it will be. (CLO-4,L1)

- a. Concurrency
- b. Semaphore
- c. Mutual exclusion
- d. parallel process

Ans:a

10. _____ is a light-weight cooperatively-scheduled execution unit.(CLO-4,L3)

- a. gevent.Greenlet
- b. gevent.spawn()
- c. gevent.spawn_later()
- d. gevent.spawn_raw()

Ans:a

11. Which keyword is used to define methods in Python? (CLO-4,L2)

- (a) function
- (b) def
- (c) method
- (d) All of these

Ans:B

12. What is the correct translation of the following statement into mathematical logic? "Some real numbers are rational" .(CLO-4,L3)

- (A) $\exists x (\text{real}(x) \vee \text{rational}(x))$
- (B) $\forall x (\text{real}(x) \rightarrow \text{rational}(x))$
- (C) $\exists x (\text{real}(x) \wedge \text{rational}(x))$
- (D) $\exists x (\text{rational}(x) \rightarrow \text{real}(x))$

A

B

C

D

Ans: c

13. Which one of the following options is CORRECT given three positive integers x, y and z, and a predicate? .(CLO-4,L3)

$$P(x) = \neg(x=1) \wedge \forall y (\exists z (x=y*z) \Rightarrow (y=x) \vee (y=1))$$

$P(x)$ being true means that x is a prime number

$P(x)$ being true means that x is a number other than 1

$P(x)$ is always true irrespective of the value of x

$P(x)$ being true means that x has exactly two factors other than 1 and x

Ans: a

14. Suppose the predicate $F(x, y, t)$ is used to represent the statement that person x can fool person y at time t. which one of the statements below expresses best the meaning of the formula $\forall x \exists y \exists t (\neg F(x, y, t))$? .(CLO-4,L3)

- (a) Everyone can fool some person at some time
- (b) No one can fool everyone all the time
- (c) Everyone cannot fool some person all the time
- (d) No one can fool some person at some time

Ans: b

15. Which one of the following is the most appropriate logical formula to represent the statement? “Gold and silver ornaments are precious”. (CLO-4,L3)

The following notations are used:

$G(x)$: x is a gold ornament

$S(x)$: x is a silver ornament

$P(x)$: x is precious

- (a) $\forall x(P(x) \rightarrow (G(x) \wedge S(x)))$
- (b) $\forall x((G(x) \wedge S(x)) \rightarrow P(x))$
- (c) $\exists x((G(x) \wedge S(x)) \rightarrow P(x))$
- (d) $\forall x((G(x) \vee S(x)) \rightarrow P(x))$

Ans : d

16. Which one of the first order predicate calculus statements given below correctly express the following English statement? .(CLO-4,L3)

Tigers and lions attack if they are hungry or threatened.

- (A) $\forall x [(\text{tiger}(x) \wedge \text{lion}(x)) \rightarrow \{(\text{hungry}(x) \vee \text{threatened}(x)) \rightarrow \text{attacks}(x)\}]$
- (B) $\forall x [(\text{tiger}(x) \vee \text{lion}(x)) \rightarrow \{(\text{hungry}(x) \vee \text{threatened}(x)) \wedge \text{attacks}(x)\}]$
- (C) $\forall x [(\text{tiger}(x) \vee \text{lion}(x)) \rightarrow \{\text{attacks}(x) \rightarrow (\text{hungry}(x) \vee \text{threatened}(x))\}]$
- (D) $\forall x [(\text{tiger}(x) \vee \text{lion}(x)) \rightarrow \{(\text{hungry}(x) \vee \text{threatened}(x)) \rightarrow \text{attacks}(x)\}]$

Ans: d

17. What is the first order predicate calculus statement equivalent to the following? (CLO-4,L3)

Every teacher is liked by some student

- (A) $\forall(x) [\text{teacher}(x) \rightarrow \exists(y) [\text{student}(y) \rightarrow \text{likes}(y, x)]]$
- (B) $\forall(x) [\text{teacher}(x) \rightarrow \exists(y) [\text{student}(y) \wedge \text{likes}(y, x)]]$
- (C) $\exists(y) \forall(x) [\text{teacher}(x) \rightarrow [\text{student}(y) \wedge \text{likes}(y, x)]]$

(D) $\forall (x) [\text{teacher}(x) \wedge \exists (y) [\text{student}(y) \rightarrow \text{likes}(y, x)]]$

Ans: b

- 18.
- I. $\neg \forall x (P(x))$ II. $\neg \exists x (P(x))$
III. $\neg \exists x (\neg P(x))$ IV. $\exists x (\neg P(x))$

Which of the above two are equivalent? (CLO-4,L3)

- (A) I and III
(B) I and IV
(C) II and III
(D) II and IV

Ans: b

19. _____ is a builtin python module where all possible types are defined. .(CLO-4,L2)

- (a) overload
b)typing
c)function
d)literal

Ans: b

20. _____ type represents a specific value of the specific type. .(CLO-4,L1)

- a) overload
b) typing
c) literal
d) None of the above

Ans: c

21. _____ is required to define multiple function declarations with different input types and results. .(CLO-4,L1)

- a) overload
- b) typing
- c) literal
- d) None of the above

Ans: a

PART B:(4 MARKS)

1. State parallel programming paradigm. (CLO-4, L1)
2. Differentiate parallel programming with functional programming. (CLO-4, L2)
3. Explain about Multithreading. (CLO-4, L1)
4. Explain about Multiprocessing. (CLO-4, L1)
5. Relate Serial processing concepts in Python. (CLO-4, L3)
6. Differentiate Serial Processing and Parallel Processing. (CLO-4, L3)
7. Demonstrate Multiprocessing module in Python. (CLO-4, L3)
8. Describe about Process class. (CLO-4, L2)
9. Design a Pool class in Python.(CLO-4, L3)
10. State Concurrent programming paradigm. (CLO-4, L1)
11. Compare multiprocessing and multitasking. (CLO-4, L2)
- 12.What are dependent functions ?(CLO4-L1)
13. Define typing module in dependent type programming? (CLO-4, L2)
- 14.Define dependent functions? (CLO-4, L2)
- 15.What is predicate logic? (CLO-4, L2)
- 16.Different types of quantifiers. (CLO-4, L1)

17.Explain Universal Quantifier and Existential Quantifier (CLO-4, L1)

18.Write some examples of Universal Qunatifier. (CLO-4, L2)

19 Define overload and literal in dependent type programming (CLO-4, L2)

20. Write the syntax for Existential Quantifier. (CLO-4, L2)

PART :C(12 MARKS)

1. Write a python program to implement producer consumer problem. (CLO-4, L3)

2. Implement the concept “Pool class” by importing a package pool. (CLO-4, L3)

3. Write a python program to implement dinning philosopher problem. (CLO-4, L3)

4. Explain the differences between multithreading and multiprocessing with example? (CLO-4, L1)

5. Write a program to implement thread synchronization (CLO-4, L3)

6. Compare Concurrent programming paradigm and functional programming paradigm with example program. (CLO-4, L2)

7. Explain in detail about dependent type programming .(CLO4-L1)

8. Write a python program to create Type aliases using typing module.. (CLO-4, L3)

9. Write a python program to check every **key:value** pair in a dictionary and check if they match the **name@email** format using typing module.. (CLO-4, L3)

10. Write a python program to create new user defined type Student using typing module.. (CLO-4, L3)

11. Write a python program for function dependent type using overload and literals .. (CLO-4, L3)

1. What does a threading.Lock do?

- a) Pass messages between threads
- b) Allow only one thread at a time to access a resource**
- c) Wait until a thread is finished
- d) SHIFT TO ALL CAPS

2. What does the Thread.join() method do?

- a) Waits for the thread to finish**
- b) Restricts access to a resource
- c) Adds the thread to a pool
- d) Merges two threads into one

3) Race conditions are ...

- a) Testing which thread completes first
- b) Two threads incorrectly accessing a shared resource**
- c) The weather on race day
- d) Something you should add to your code

4) It sets the lock state to locked. If called on a locked object, it blocks until the resource is free.

- a) lock()
- b)release()
- c) acquire()**
- d)join()

5) You have thread T1, T2, and T3. How will you ensure that thread T2 is run after T1 and thread T3 after T2?

- a) Sleep method
- b) Join Method**
- c)Release
- d)lock method

6) What are the libraries in Python that support threads?

- _threading
- b) Thread
- c)None
- d)thread**

7) Mention the correct syntax for creating Thread object for calling increment methods

- a) t1 = threading.Thread()
- b) t1 = threading.Thread(target)
- c) t1 = threading.Thread(target=incr)**
- d) t1 = threading.Thr

8) Which leads to concurrency?

- a) Serialization
- b) Parallelism**
- c) Serial processing
- d) Distribution

9) Which is not a method for parallelism?

- a) Message Passing
- b) Shared Memory
- c) Threads
- d) Sockets**

10) A race condition occurs when multiple processes or threads read and write

- a). Input b). Information c). **Data Items** d). Programs

11) For a single processor system, implementation of semaphores is possible to inhibited through

- a) Deadlock b) **Interrupts** c) Lock Step d) Paging

12) Which method controls the execution of thread in python?

- a) Wait b) **Sleep** c) Acquire d) Lock

13) How does run() method is invoked?

- a) By Thread.create() b) **By Thread.start()**

- c) None d) By Thread.run()

14) What is the difference between *threading.Lock* and *threading.RLock*?

- a) Lock and RLock both primitives are owned by a single thread.

- b) **Lock is owned by none while RLock is owned by many.**

- c) Lock is owned by a thread while RLock is owned by many.

- d) Lock and RLock both primitives are owned by many.

15) What is the difference between a *semaphore* and *bounded semaphore*?

- a) Semaphore holds a counter for the number of release() calls minus the number of acquire() calls, plus an initial value but bounded semaphore doesn't.

- b) A semaphore makes sure its current value doesn't exceed its initial value while bounded semaphore doesn't.

- c) **A bounded semaphore makes sure its current value doesn't exceed its initial value while semaphore doesn't.**

- d) Bounded semaphore holds a counter for the number of release() calls minus the number of acquire() calls, plus an initial value but semaphore doesn't.

16) What is the method that wakes up all thread waiting for the condition?

- a) releaseAll() b) notify() c) **notifyAll()** d) release()

17 How to terminate a blocking thread?

- a). **thread.stop() & thread.wait()** b) thread.stop()

- c) thread.terminate() d) None

18. Which synchronization method is used to guard the resources with limited capacity, e.g. a database server?

- a) Event b) Condition c) Lock d) **Semaphore**

19. How to detect the status of a python thread?

- a) isActive()
- b) isDaemon()
- c) None
- d) **isAlive()**

20. Execution of several activities at the same time.

- a) processing
- b) parallel processing**
- c) serial processing
- d) multitasking

1. SymPy is a collection of mathematical algorithms and convenience functions built on the [-----extension of Python](#)

- a) **numpy**
 - b) **scikit**
 - c) **sys**
 - d) **functools**

2. Exponential function computes the -----

- a) $10^{**}x$ element-wise
 - b) $10^{**}x$ row-wise
 - c) $10^{**}x$ column-wise
 - d) 10^*x element-wise

3. _____ evaluates the expression to a floating-point number.

- a) evalf
 - b) fval
 - c) float
 - d) valf

4. what is the output for the following expression `((x+y)**2).expand()`

- a) $x, 2 + 2, x, y + y, 2$ b) $x^2 + 2 \cdot x \cdot y + y^2$

5. `limit((5**x + 3**x)**(1/x), x, oo)`, what is the output

6) Higher derivatives can be calculated using the which method

- a) `highder(func,var,n)` b) `diff(func, var, n)`
c) `diff(n,var,func)` d) `diff(func, var)`

7) what is the output

```
>>> x = Symbol('x')
```

```
>>> y = Symbol('y')
```

```
>>> A = Matrix([[1,x], [y,1]])
```

```
>>> A**2
```

- a) $[1, x]$
 $[y, 1]$

b) $[xy + 1, 2x]$
 $[2y, xy + 1]$

[$2*x$, $x*y + 1$]

[$2*y$, $x*y + 1$]

8) .match() method, along with the ----- class, to perform pattern matching on expressions.

- a) pattern
- b) func
- c) wild**
- d) dictionary

9) which among the following function Return or print, respectively, a pretty representation of expr

- a) pretty(expr)
- b) pretty_print(expr)
- c) pprint(expr)
- d) all of the above**

10) What is the output of

```
from sympy.abc import a, b
```

```
expr = b*a + -4*a + b + a*b + 4*a + (a + b)*3
```

- a) $ba-4a+b+ab+4a+3(a+b)$
- b) $2*a*b + 3*a + 4*b$**
- c) $2ab+3a+4b$
- d) all of the above

11) print(pi.evalf(30))

- a) $3.14/30$
- b) $30/3.14$
- c) 3.14159265358979323846264338328**
- d) 3.14

12) which is the correct way to write equation for $x^2=x$ in sympy

- a) $x^{**2} = x$**
- b) $x*x = x$
- c) $x\%2 = x$
- d) $X^2 = x$

13) how to find a solution for a equation in a given interval

- a) solve(equation,range)
- b) equation(solve,range)
- c) solveset()**
- d) both a and b

14) Allows , the same elements can appear multiple times at different positions

a) set **b)sequence**

c) dictionary d) none

15) which is the snippet to find the eigenvalues of $\begin{bmatrix} 1 & 2 \\ 2 & 2 \end{bmatrix}$

- a) `Matrix([[1, 2], [2, 2]]).eigenvals()`
b) `Matrix([[1, 2], [2, 2]]).eigenvalues()`
c) both a and b d) `eigen([[1, 2], [2, 2]])`

16 What is the purpose of sympify() method?

- a. Convert expression of string type to mathematical expression
- b. Convert mathematical expression to String
- c. Convert mathematical expression to character
- d. Convert tuple to mathematical expression

17 Find the output of the following program

```
from sympy import solve
x = Symbol('x')
expr = x**2 + 5*x + 4
solve(expr, dict=True)
```

- a. $\{x: -4\}, \{x: -1\}$
- b. $\{x: -6\}, \{x: -1\}$
- c. $\{x: -1\}, \{x: -4\}$
- d. $\{x: 4\}, \{x: -1\}$

18. A *rational expression* is an algebraic expression in which the numerator and denominator are both -----

- a. Equal
- b. **Polynomials**
- c. Unequal
- d. Symmetric

19. Find the output of the following program

```
# import sympy
from sympy import *
```

```

x = symbols('x')
expr = sin(x)/x;
print("Expression : {}".format(expr))

# Use sympy.limit() method
limit_expr = limit(expr, x, 0)
print("Limit of the expression tends to 0 : {}".format(limit_expr))

```

- a. Expression : $\cos(x)/x$
Limit of the expression tends to 0 : 2
- b. Expression : $\tan(x)/x$
Limit of the expression tends to 0 : 3
- c. Expression : $\sin(x)/x$
Limit of the expression tends to 1 : 0
- d. **Expression : $\sin(x)/x$**
Limit of the expression tends to 0 : 1

20. ----- method will simplify mathematical expression using trigonometric identities.

- a. **sympy.trigsimp()**
- b. sympy.series()
- c. sympy.lambda()
- d. sympy.sim()

Which of the following programming paradigms allow us to write programs and know they are correct before running them?

- a) Automata based Programming Paradigm
- b) Logical Programming Paradigm
- c) **Dependent type Programming Paradigm**
- d) Imperative Programming Paradigm

2) Which of the following is false regarding dependent types?

- a) They allow us to write programs and know they are correct before running them.
- b) **They allow us to write programs and know they are correct only after running them.**
- c) You can specify types that can check the value of your variables at compile time.
- d) Its definition depends on a value.

3) Which of the notations is true for “P(x) is true for all values of x in the universe of discourse”?

- a) $x\forall P(x)$
- b) $\forall P(x)x$
- c) $\exists xP(x)$
- d) **$\forall xP(x)$**

4) Which of the following is false about quantifiers?

- a) Notation \forall is used for Universal quantifier.
- b) Notation \exists is used for Existential quantifier.
- c) “All men drink tea” is an example of Universal Quantifier
- d) **“All men drink coffee” is an example of Existential Quantifier.**

5) Let P(x) be the predicate “x must take a discrete mathematics course” and let Q(x) be the predicate “x is a computer science student”.

Which of the following statements is correct for “Everybody must take a discrete mathematics course or be a computer science student”?

- a) **$\forall x(Q(x) \vee P(x))$**
- b) $\forall x(Q(x)) \vee \forall x(P(x))$
- c) $\forall x(Q(x) \parallel P(x))$
- d) $\forall x(Q(x) \rightarrow P(x))$

6) Which of the following is correct for predicate “All men drink coffee”?

- a) $\forall x \text{men}(x) \rightarrow (x, \text{coffee})$
- b) $\text{drink}(x, \text{coffee}) \rightarrow \forall x \text{men}(x)$

- c) $\forall x \text{ men}(x) \rightarrow \text{drink}(x, \text{coffee})$
d) $\forall \text{men}(x) \rightarrow \text{drink}(x, \text{coffee})$
- 7) Which of the following is correct for predicate "Some boys play football"?
a) $\exists x \text{ boys}(x) \rightarrow \text{play}(x, \text{football})$
b) $\exists \text{ boys}(x) \rightarrow \text{play}(x, \text{football})$
c) $\forall x \text{ boys}(x) \rightarrow \text{play}(x, \text{football})$
d) $\exists x \forall \text{boys}(x) \rightarrow \text{play}(x, \text{football})$
- 8) Dependent Type is used to encode _____ like "for all" and "there exists".
a. **Logic Quantifiers**
b. Analysing Quantifiers
c. Dependent Quantifiers
d. None of the above
- 9) "There exists x in the universe of discourse such that P(X) is true" is which Quantifiers statement?
a. Logical
b. Universal
c. **Existential**
d. None of these
- 10) What is the way to determine if a given function is dependant type
a. Result is independent of the argument
b. Result depends upon the usage in the program
c. **Result depends on the Value of its argument**
d. Result depends on available resources
- 11) Notation for an Existential Quantifier:
a. $\forall x P(x)$
b. $\Sigma P(x)$
c. $\emptyset x P(x)$
d. **$\exists x P(x)$**
- 12) Representation of the following statement:
Every Clock has quartz
a. $\emptyset x \text{clock}(x) \rightarrow \text{quartz}(x)$
b. $\exists x \text{clock}(x) \rightarrow \text{quartz}(x)$
c. **$\forall x \text{clock}(x) \rightarrow \text{quartz}(x)$**
d. none of the above
- 13) Representation of the following statement:
Some leaves are Red

a. $\exists x \text{leaves}(x) \rightarrow \text{red}(x)$

b. $\forall x \text{leaves}(x) \rightarrow \text{red}(x)$

c. $\Sigma \text{leaves}(x) \rightarrow \text{red}(x)$

d. none of the above

13. A function has dependent type if the _____ of a function's result depends on the _____ of its Argument

a. value and type

b. type and value

c. type and type

d. value and value

14. Dependent type paradigm used to encode logic's quantifiers like _____ and _____

a. for one, there may exists

b. for all, there always

c. for all, there exists

d. for specific, there exists

15. Choose the correct one with respect to typing and typing-extensions library in python dependent type programming.

a. typing is not builtin python module where all possible types are defined.

typing_extensions is an official package for new types in the future releases of python

b. typing is a builtin python module where all possible types are defined.

typing_extensions is an official package for new types in the future releases of python

c. typing is a builtin python module where all possible types are defined.

typing_extensions is not an official package for new types in the future releases of python

d. typing is not a builtin python module where all possible types are defined.

typing_extensions is not an official package for new types in the future releases of python

16. Predict the output of the below mentioned python code without dependent type syntax:

```
def make_hamburger(meat, number_of_meats):
    return ["bread"] + [meat] * number_of_meats + ["bread"]
print(make_hamburger("ground beef", 2))
```

a. ['bread', 'ground beef', 2, 'bread']

b. ['bread', 'ground beef', 'ground beef', 'bread']

c. *TypeError: cannot concatenate 'str' and 'int' objects*

d. `['bread', 'ground beef', '2bread']`

17. Predict the output of the below mentioned python code with dependent type syntax:

```
from typing import List

def greet_all(names: List[str]) -> None:
    for name in names:
        print('Hello ' + name)

names = ["Alice", "Bob", "Charlie"]
ages = [10, 20, 30]

greet_all(names)
greet_all(ages)

a. greet_all(names) # Ok!
    greet_all(ages) # Ok!

b. a.greet_all(names) # Ok!
    greet_all(ages) # Error due to incompatible types

c. a.greet_all(names) # Error due to incompatible types
    greet_all(ages) # Ok!

d. a.greet_all(names) # Error due to incompatible types
    greet_all(ages) # Error due to incompatible types
```

18 Let x be a variable which refers to Universe of Disclosure such as x1,x2....xn then

,how to represent this statement using quantifiers “ All Man working in Industry”

- a) $\forall x \text{ man}(x) \rightarrow \text{work}(x, \text{Industry})$ b) $\forall x \text{ man}(x) \rightarrow \text{work}(\text{industry})$.
- a) $\forall x \rightarrow \text{work}(x, \text{industry})$ d) $\forall x \text{ man}(x) \rightarrow \text{Industry}(\text{work})$

19 How do you represent the statement “Every man respects his parent.”

- a) $\forall x \text{ woman}(x) \rightarrow \text{respects}(x, \text{parent})$ b) $\forall x \text{ man}(x) \rightarrow \text{respects}(x, \text{parent})$
- c) $\forall x \text{ man} \rightarrow \text{respects}(x, \text{parent})$ d) $\forall x \text{ man}(x) \rightarrow \text{respects}(\text{parent})$.

20 How do you represent the statement “Some boys play cricket “

- a) $\exists x \text{ boys}(x) \rightarrow \text{play}(\text{all})$ b) $\forall x \text{ boys}(x) \rightarrow \text{play}(x, \text{cricket})$
- c) $\exists x \text{ boys}(x) \rightarrow \text{play}(x, \text{cricket})$ d) $\exists x \wedge \forall x \text{ boys}(x) \rightarrow \text{play}(x, \text{cricket})$

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
VADAPALANI CAMPUS
DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

18CSC207J – ADVANCED PROGRAMMING PRACTICES

NETWORK PROGRAMMING PARADIGM

1. _____ programming is a way of connecting two nodes on a network to communicate with each other.
 - i. Logic
 - ii. **Socket**
 - iii. Functional
 - iv. Symbolic

2. Which method used by the server to initiate the connection with the client?
 - i. Listen()
 - ii. Close()
 - iii. Bind()
 - iv. **Accept()**

3. _____ is a socket through which data can be transmitted continuously.
 - i. Datagram Socket
 - ii. **Stream Socket**
 - iii. Raw Socket
 - iv. Binary Socket

4. _____ paradigm deals with client – server communication
 - i. Dependent type paradigm
 - ii. Parallel programming paradigm
 - iii. Network paradigm
 - iv. Concurrent programming paradigm

5. _____ protocol facilitates sending of datagrams in an unreliable manner.

- i. TCP
 - ii. UDP**
 - iii. HTTP
 - iv. FTP
- 6. Which among methods are not server socket?
 - i. connect()**
 - ii. bind()
 - iii. listen()
 - iv. accept()
- 7. In UDP, Which among methods are used to receive messages at endpoint
 - i. sock_object.recv()
 - ii. sock_object.send()
 - iii. sock_object.recvfrom()**
 - iv. sock_object.sendto()
- 8. In TCP, Which among methods are used to send messages from endpoint
 - i. sock_object.recv()
 - ii. sock_object.send()**
 - iii. sock_object.recvfrom()
 - iv. sock_object.sendto()
- 9. The protocol which defines IPv4 is
 - i. AF_UNIX
 - ii. SOCK_STREAM
 - iii. AF_INET**
 - iv. SOCK_DGRAM
- 10. The correct order of methods used in server socket is
 - i. Socket(), Bind(), listen(), accept()**
 - ii. Socket(), listen(), bind(), accept()
 - iii. Socket(), accept(), bind(), listen()
 - iv. Socket(), bind(), accept(), listen()

11. _____ is a type of network socket which provides connection less point for sending and receiving packets.

- i. **Datagram Socket**
- ii. Stream Socket
- iii. Raw Socket
- iv. Binary Socket

12. _____ do not use any transport protocol but data is directly transmitted over IP protocol

- i. Datagram Socket
- ii. Stream Socket
- iii. **Raw Socket**
- iv. Binary Socket

13. The _____ is a physical path over which the message travels.

- i. Protocol
- ii. **Medium**
- iii. Path
- iv. Route

14. A pair (host, port) is used for the _____ address family.

- i. AF_NETLINK
- ii. AF_INET6
- iii. **AF_INET**
- iv. AF_ALG

15. Use _____ to make the socket to visible to the outside world.

- i. socket.listen()
- ii. socket.visible()
- iii. socket.socket()
- iv. **Socket.gethostname()**

16. In which mode socket is created in default.

- i. **Blocking mode**
- ii. Non-Blocking Mode
- iii. Timeout mode
- iv. Accept mode

17. Which method is recommended to be called before calling connect() method?

- i. getdefaulttimeout()

- ii. **settimeout()**
- iii. **getaddrinfo()**
- iv. no such method

18. Which exception is raised for address related errors by getaddrinfo()?

- i. gaoerror
- ii. gsierror
- iii. **gaierror**
- iv. gdeerror

19. _____ protocol facilitates sending of datagrams in an reliable manner.

- i. **TCP**
- ii. UDP
- iii. HTTP
- iv. FTP

20. To create a socket, which function among the following is available in python socket module?

- i. socket.create()
- ii. socket.initialize()
- iii. **socket.socket()**
- iv. socket.build()

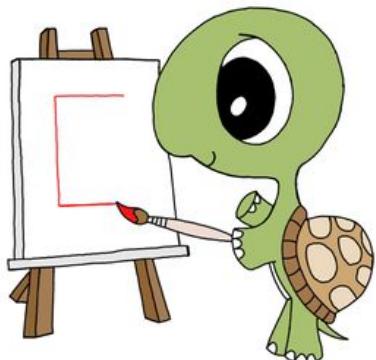


Event-Driven Programming

Using Turtle

Ref:

1. How to Think Like a Computer Scientist: Learning with Python
3
2. Review of Programming Paradigms Throughout the History
With a Suggestion Toward a Future Approach



Event- Driven Programming

- A dominant paradigm used in **Graphical User Interface (GUI)**
- Flow of the programs in GUI is determined by user actions also known as events
 - Mouse clicks, key presses, sensor outputs and so on
- Event driven applications use a loop to detect events
- Occurrence of an event triggers a function called as Event Handlers
- Event-driven programs can be easily developed in all object-oriented languages and Visual languages comparatively.
 - Eg.: Visual Basic, Visual C++, Java, VB.Net

How Event-Driven Programming works?

- **Event Scheduler** is central to event-driven programming
- Scheduler receives an event and passes the event to its respective handler
- Each event has an Unique ID and additional information such as, x and y coordinates of mouse pointer in a mouse event or state of shift key in key press event
- **How Events are generated?**
 - Actions performed by user during the execution of a program like mouse click, pressing a button, selecting a choice button or a radio button
 - Messages generated by peripheral device, System hardware
 - Completion of file download generates an event

Event Handlers

- A block of code that executes when an event occurs.
- Produces a visual response to inform/direct the user
- Changes the system's state
 - System state includes data used by the system and the state of the user interface-on-screen object has the focus
 - Sometimes event handler calls another event handler.
- Event handlers are linked to the on screen object to which it should listen-called as binding

Event-Driven Programming

- Python is used to demonstrate Event-driven programs
- Libraries that support Event driven programming in Python are
 - Turtle
 - Tkinter
 - PyQt5
 - WxWidgets

Turtle

- Turtle is a library included in Python
- Turtle is used to draw shapes and patterns and handle events
- Turtle module is imported

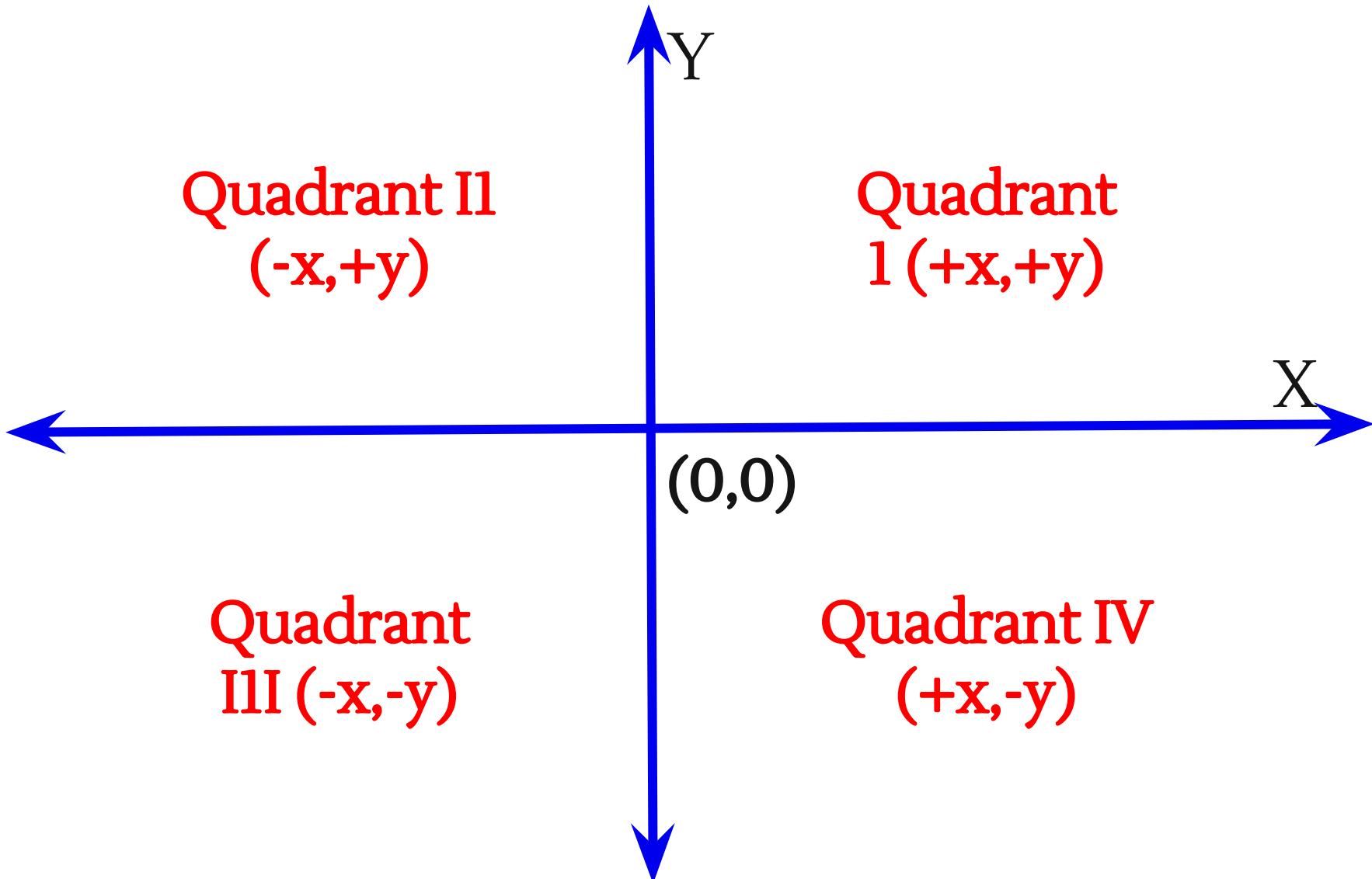
```
import turtle * or from turtle import *
```

- Following command opens a window called as screen and a black triangle in the middle called turtle

```
s = turtle.getscreen()
```

- Screen acts as a canvas and turtle acts as a pen
- Turtle has specific characteristics like, size, colour, speed and direction in which the turtle can move

Turtle Screen



Turtle commands

- Movement
 - `forward(x)/ fd(x)`-moves turtle in forward direction by x units
 - `backward(x)/bk(x)/back(x)`-moves turtle in backward by x units
 - `left(d)/lt(d)`-turns the turtle left by d degrees
 - `right(d)/rt(d)`-turns the turtle right by d degrees

Other commands

- goto() | setpos() |
setposition()
- setx()
- sety()
- setheading() | seth()
- home()
- circle()
- dot()
- stamp()
- clearstamp()
- clearstamps()
- undo()
- speed()
- And many more such
commands are available

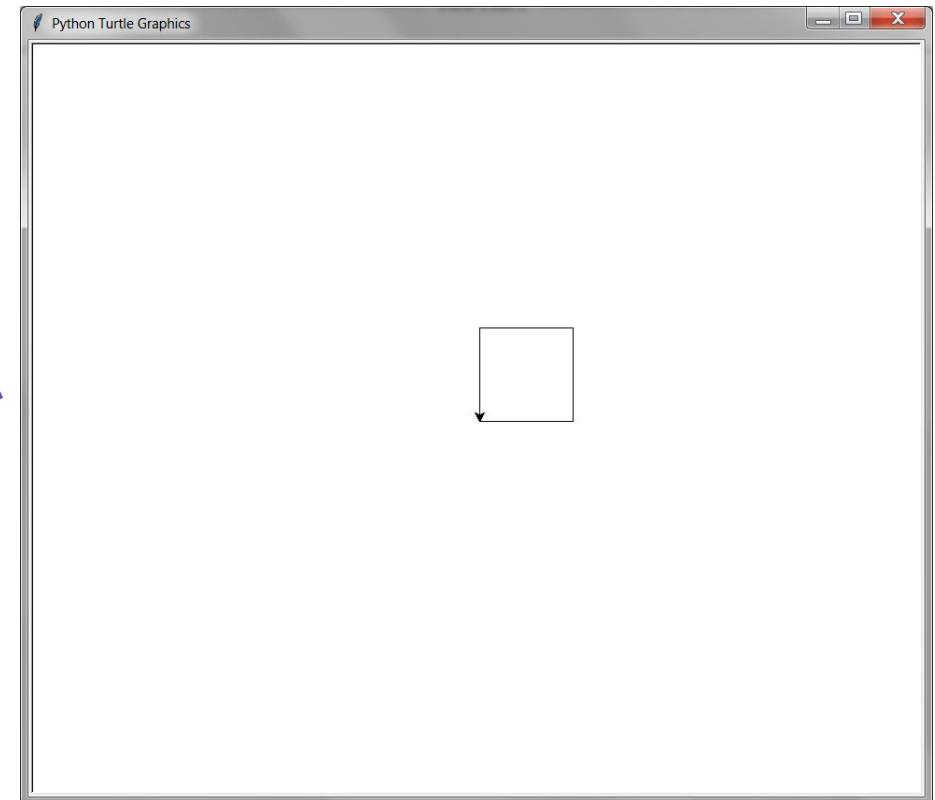
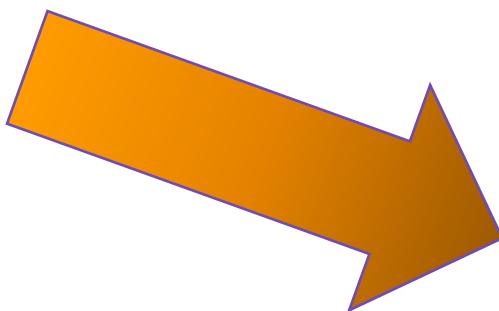
Event handling commands

- onclick()
- onrelease()
- ondrag()
- listen()
- onkey() | onkeyrelease()
- onkeypress()
- onclick() | onscreenclick()
- ontimer()
- mainloop() | done()

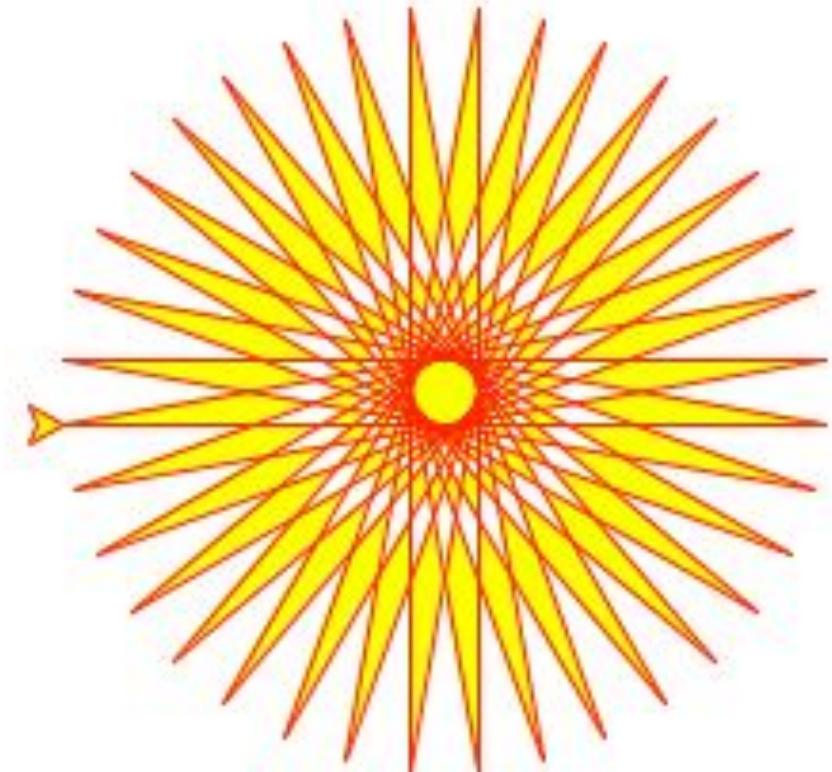
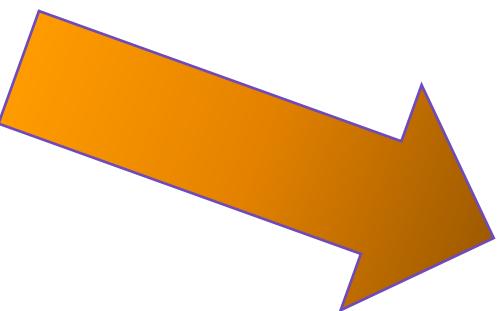
Simple programs using turtle

```
from turtle import *
```

```
forward(100)  
left(90)  
forward(100)  
left(90)  
forward(100)  
left(90)  
forward(100)
```



```
from turtle import *
color('red', 'yellow')
begin_fill()
while True:
    forward(200)
    left(170)
    if abs(pos()) < 1:
        break
end_fill()
done()
```

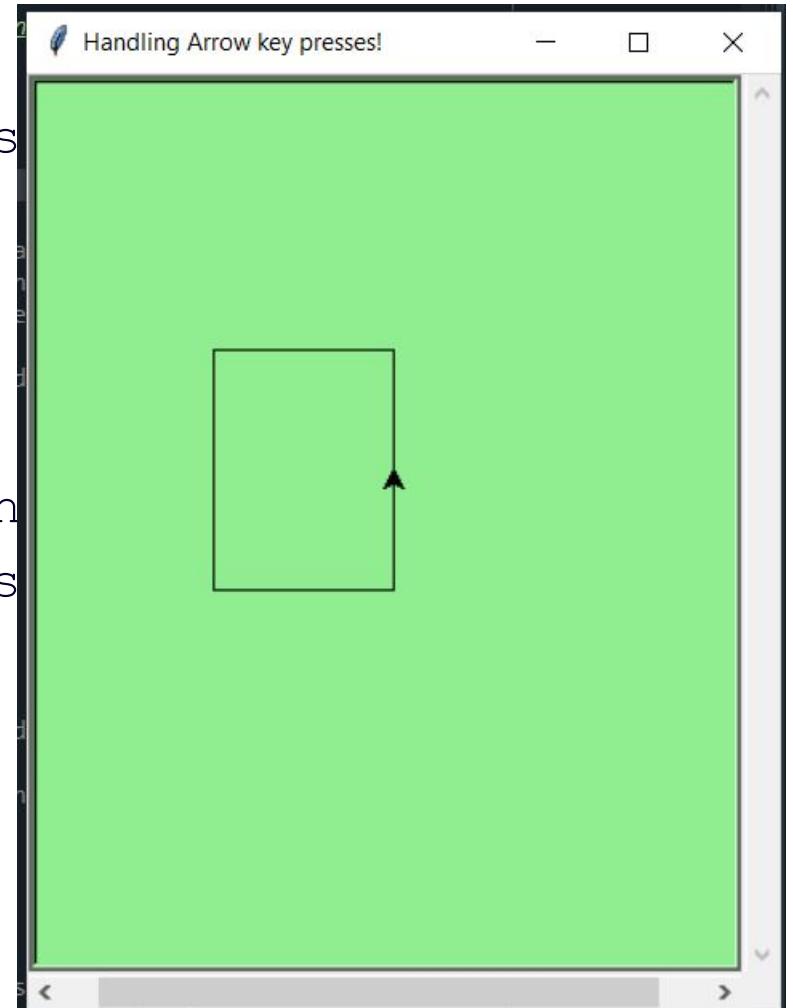
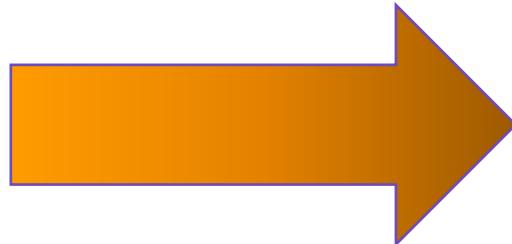


Keypress event

```
import turtle  
turtle.setup(400,500,10,20)      # Determine the window size  
wn = turtle.Screen()            # Get a reference to the window  
wn.title("Handling keypresses!") # Change the window title  
wn.bgcolor("lightgreen")        # Set the background color  
tess = turtle.Turtle()          # Create our favorite turtle  
# The next four functions are our "event handlers".  
def h1():  
    tess.forward(30)  
def h2():  
    tess.left(45)  
def h3():  
    tess.right(45)
```

Contd...

```
def h4():
    wn.bye()          # Close down the turtle window
# These lines "wire up" keypresses to the handlers
wn.onkey(h1, "Up")
wn.onkey(h2, "Left")
wn.onkey(h3, "Right")
wn.onkey(h4, "q")
# Now we need to tell the window to start listening
# If any of the keys that we're monitoring is pressed
# handler will be called.
wn.listen()
wn.mainloop()
```



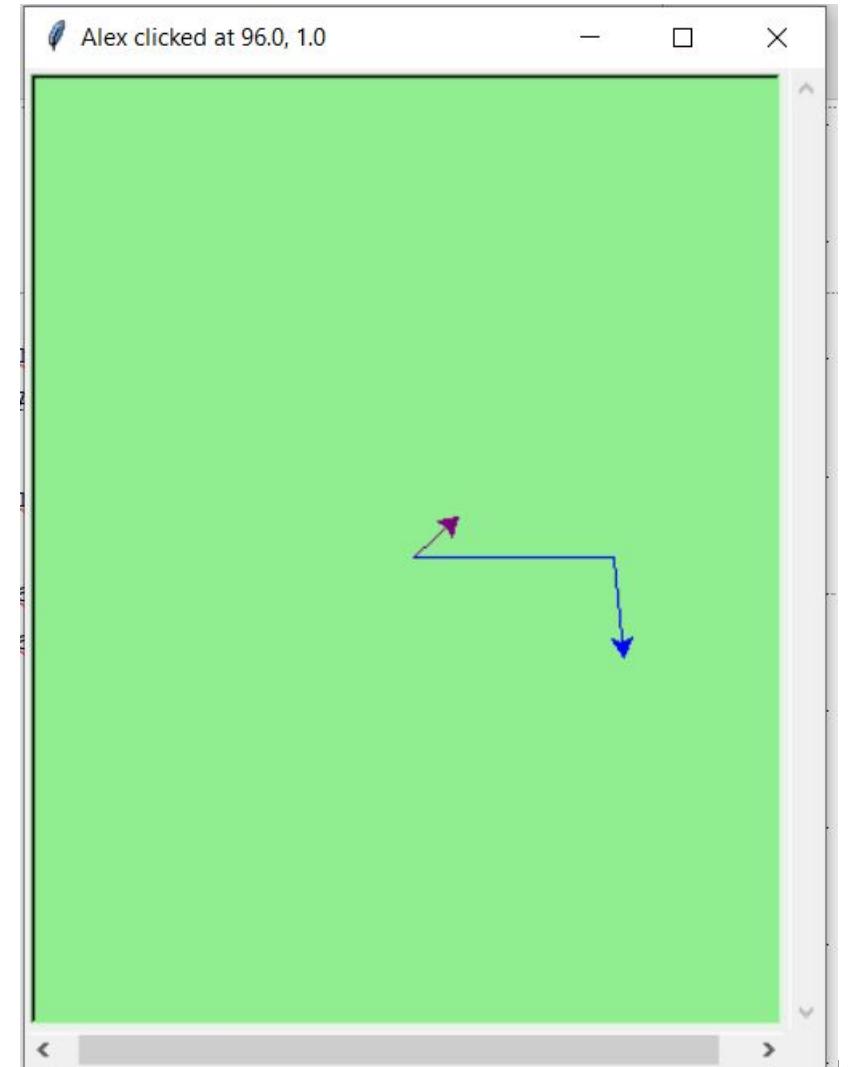
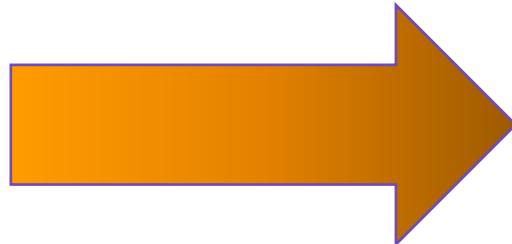
Mouse Event

```
import turtle
turtle.setup(400,500,10,20)          # Determine the window size
wn = turtle.Screen()                 # Get a reference to the window
wn.title("Handling mouse clicks!")  # Change the window title
wn.bgcolor("lightgreen")             # Set the background color
tess = turtle.Turtle()               # Create two turtles
tess.color("purple")
alex = turtle.Turtle()               # Move them apart
alex.color("blue")
alex.forward(100)

def handler_for_tess(x, y):
    wn.title("Tess clicked at {0}, {1}".format(x, y))
    tess.left(42)
    tess.forward(30)
```

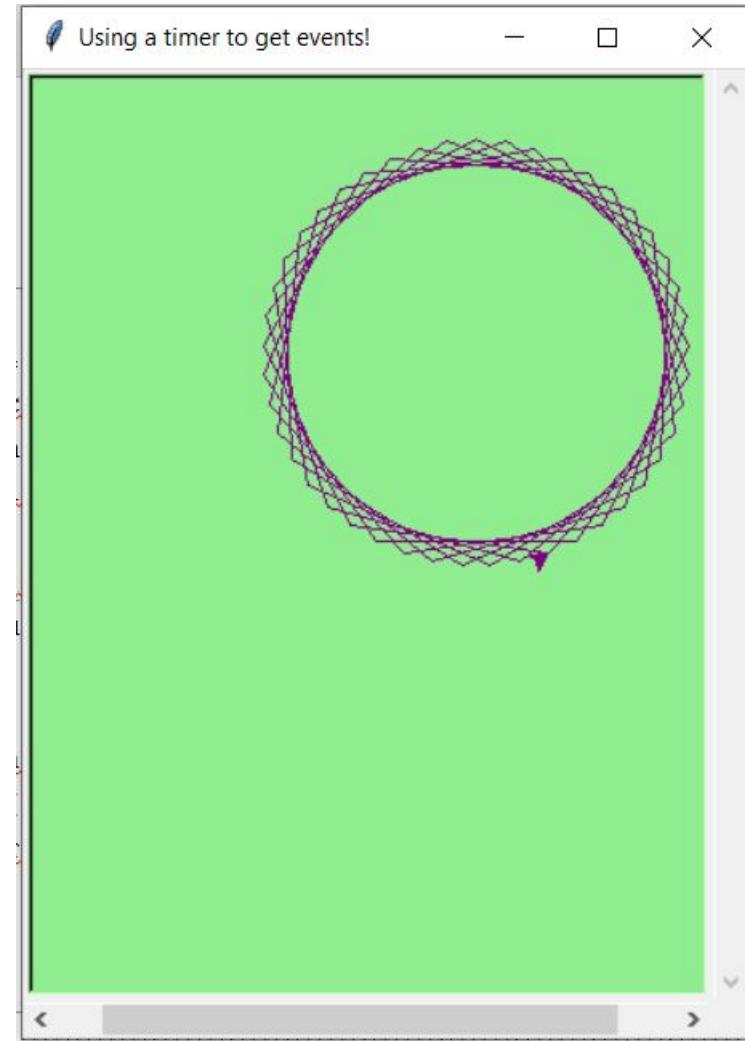
Contd...

```
def handler_for_alex(x, y):  
    wn.title("Alex clicked at {0}, {1}".format(x, y))  
    alex.right(84)  
    alex.forward(50)  
  
tess.onclick(handler_for_tess)  
alex.onclick(handler_for_alex)  
  
wn.mainloop()
```



Automatic event from Timer

```
import turtle  
turtle.setup(400,500)  
wn = turtle.Screen()  
wn.title("Using a timer to get events!")  
wn.bgcolor("lightgreen")  
  
tess = turtle.Turtle()  
tess.color("purple")  
  
def h1():  
    tess.forward(100)  
    tess.left(56)  
    wn.ontimer(h1, 60)  
  
h1()  
wn.mainloop()
```



References

- https://github.com/asweigart/simple-turtle-tutorial-for-python/blob/master/simple_turtle_tutorial.md
- <https://docs.python.org/3/library/turtle.html>
- <https://openbookproject.net/thinkcs/python/english3e/events.html#:~:text=The%20turtle%20module%20in%20Python,when%20its%20time%20is%20up.&text=On%20line%202016%20the%20timer,~and%20tess%20springs%20into%20action.>

18CSC207J – Advanced Programming Practice

GUI & Event Handling Programming Paradigm

Event Driven Programming Paradigm

- Event-driven programming is a programming paradigm in which the flow of program execution is determined by events - for example a user action such as a mouse click, key press, or a message from the operating system or another program.
- An event-driven application is designed to detect events as they occur, and then deal with them using an appropriate event-handling procedure.
- In a typical modern event-driven program, there is no discernible flow of control. The main routine is an event-loop that waits for an event to occur, and then invokes the appropriate event-handling routine.
- Event callback is a function that is invoked when something significant happens like when click event is performed by user or the result of database query is available.

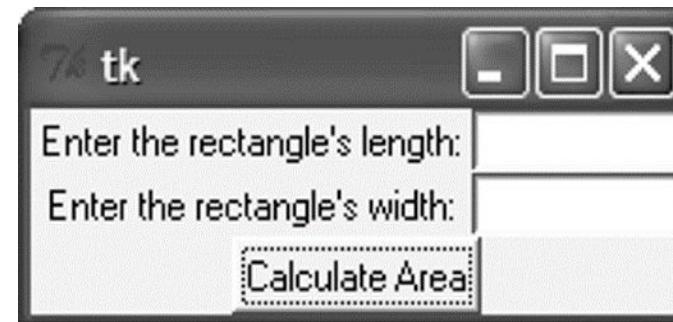
Event Handlers: Event handlers is a type of function or method that run a specific action when a specific event is triggered. For example, it could be a button that when user click it, it will display a message, and it will close the message when user click the button again, this is an event handler.

Trigger Functions: Trigger functions in event-driven programming are a functions that decide what code to run when there are a specific event occurs, which are used to select which event handler to use for the event when there is specific event occurred.

Events: Events include mouse, keyboard and user interface, which events need to be triggered in the program in order to happen, that mean user have to interacts with an object in the program, for example, click a button by a mouse, use keyboard to select a button and etc.

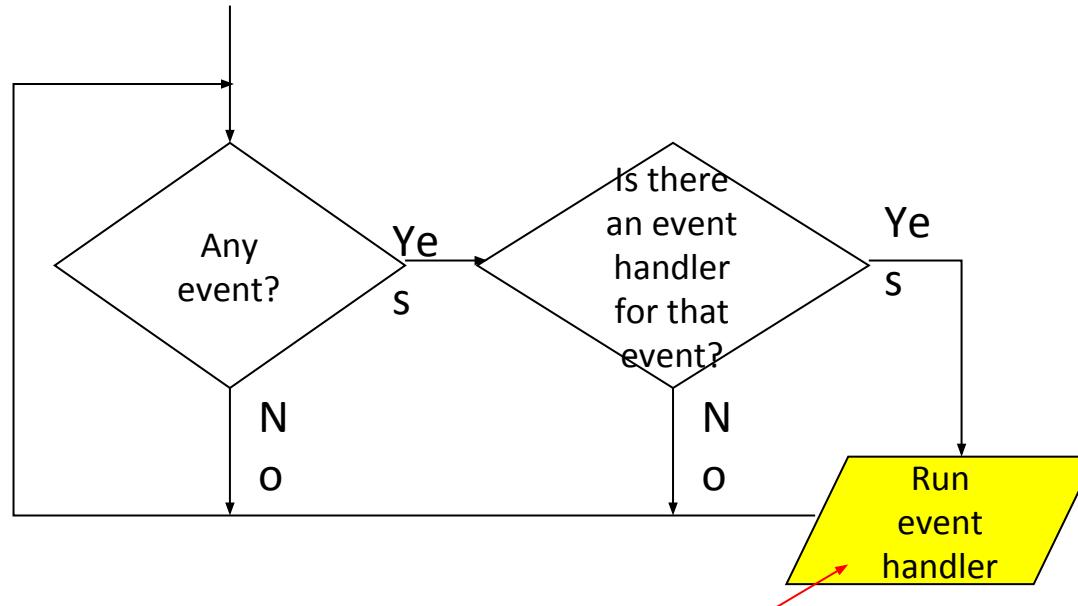
Introduction

- A graphical user interface allows the user to interact with the operating system and other programs using graphical elements such as icons, buttons, and dialog boxes.
- GUIs popularized the use of the mouse.
- GUIs allow the user to point at graphical elements and click the mouse button to activate them.
- GUI Programs Are Event-Driven
- User determines the order in which things happen
- GUI programs respond to the actions of the user, thus they are event driven.
- The tkinter module is a wrapper around tk, which is a wrapper around tcl, which is what is used to create windows and graphical user interfaces.



Introduction

- A major task that a GUI designer needs to do is to determine what will happen when a GUI is invoked
- Every GUI component may generate different kinds of “events” when a user makes access to it using his mouse or keyboard
- E.g. if a user moves his mouse on top of a button, an event of that button will be generated to the Windows system
- E.g. if the user further clicks, then another event of that button will be generated (actually it is the click event)
- For any event generated, the system will first check if there is an event handler, which defines the action for that event
- For a GUI designer, he needs to develop the event handler to determine the action that he wants Windows to take for that event.



GUI Using Python

- Tkinter: Tkinter is the Python interface to the Tk GUI toolkit shipped with Python.
- wxPython: This is an open-source Python interface for wxWindows
- PyQt –This is also a Python interface for a popular cross-platform Qt GUI library.
- JPython: JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine

Tkinter Programming

- Tkinter is the standard GUI library for Python.
- Creating a GUI application using Tkinter

Steps

- Import the Tkinter module.

Import tkinter as tk

- Create the GUI application main window.

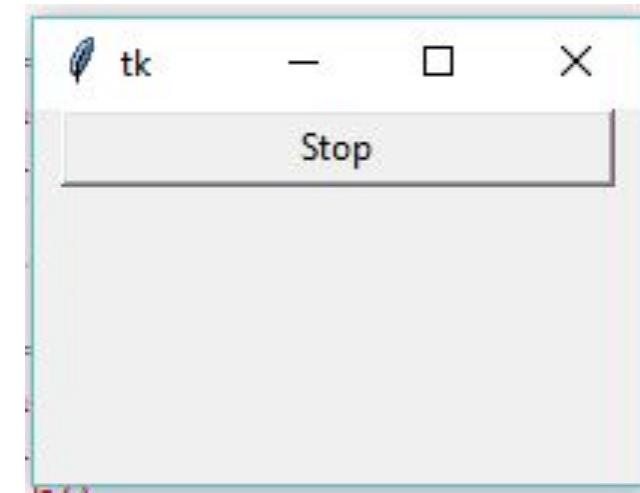
root = tk.Tk()

- Add one or more of the above-mentioned widgets to the GUI application.

*button = tk.Button(root, text='Stop', width=25, command=root.destroy)
button.pack()*

- Enter the main event loop to take action against each event triggered by the user.

root.mainloop()

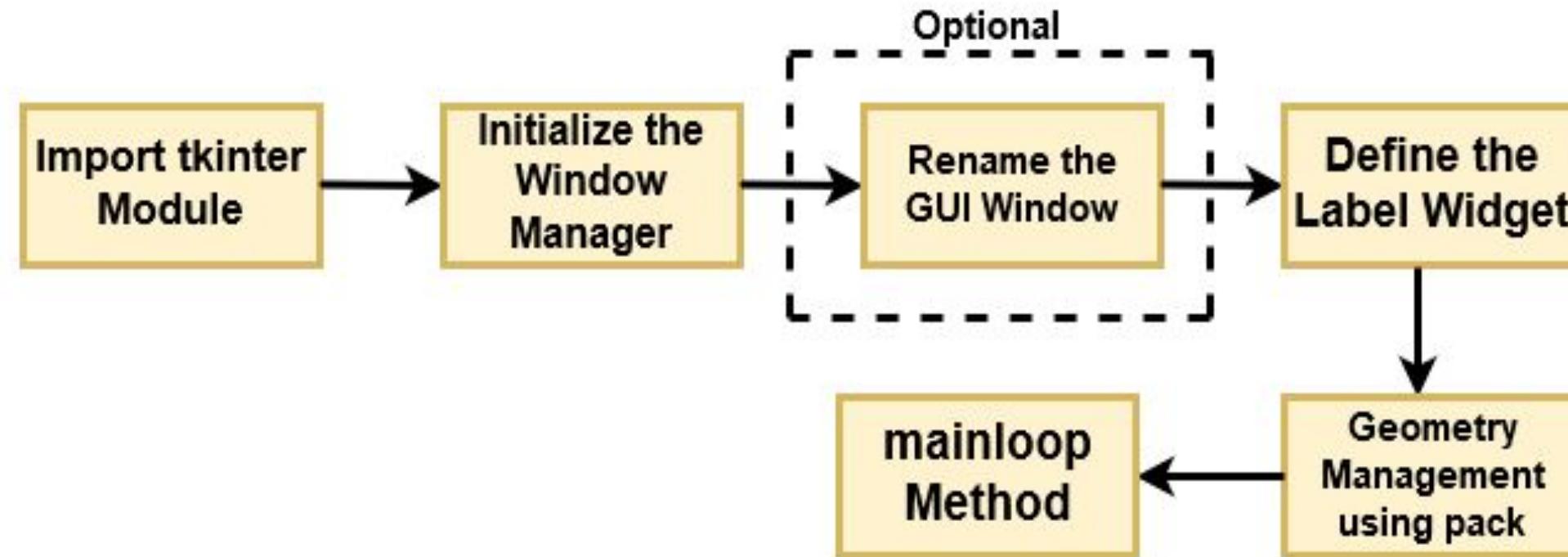


Tkinter widgets

- Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

Widget	Description
Label	Used to contain text or images
Button	Similar to a Label but provides additional functionality for mouse overs, presses, and releases as well as keyboard activity/events
Canvas	Provides ability to draw shapes (lines, ovals, polygons, rectangles); can contain images or bitmaps
Radiobutton	Set of buttons of which only one can be "pressed" (similar to HTML radio input)
Checkbutton	Set of boxes of which any number can be "checked" (similar to HTML checkbox input)
Entry	Single-line text field with which to collect keyboard input (similar to HTML text input)
Frame	Pure container for other widgets
Listbox	Presents user list of choices to pick from
Menu	Actual list of choices "hanging" from a Menubutton that the user can choose from
Menubutton	Provides infrastructure to contain menus (pulldown, cascading, etc.)
Message	Similar to a Label, but displays multi-line text
Scale	Linear "slider" widget providing an exact value at current setting; with defined starting and ending values
Text	Multi-line text field with which to collect (or display) text from user (similar to HTML TextArea)
Scrollbar	Provides scrolling functionality to supporting widgets, i.e., Text, Canvas, Listbox, and Entry
Toplevel	Similar to a Frame, but provides a separate window container

Operation Using Tkinter Widget



Geometry Managers

- The pack() Method – This geometry manager organizes widgets in blocks before placing them in the parent widget.

`widget.pack(pack_options)`

options

- expand – When set to true, widget expands to fill any space not otherwise used in widget's parent.
 - fill – Determines whether widget fills any extra space allocated to it by the packer, or keeps its own minimal dimensions: NONE (default), X (fill only horizontally), Y (fill only vertically), or BOTH (fill both horizontally and vertically).
 - side – Determines which side of the parent widget packs against: TOP (default), BOTTOM, LEFT, or RIGHT.
- The grid() Method – This geometry manager organizes widgets in a table-like structure in the parent widget.

`widget.grid(grid_options)`

options –

- Column/row – The column or row to put widget in; default 0 (leftmost column).
- Columnspan, rowsapn – How many columns or rows to widget occupies; default 1.
- ipadx, ipady – How many pixels to pad widget, horizontally and vertically, inside widget's borders.
- padx, pady – How many pixels to pad widget, horizontally and vertically, outside v's borders.

Geometry Managers

- The place() Method – This geometry manager organizes widgets by placing them in a specific position in the parent widget.

widget.place(place_options)

options –

- anchor – The exact spot of widget other options refer to: may be N, E, S, W, NE, NW, SE, or SW, compass directions indicating the corners and sides of widget; default is NW (the upper left corner of widget)
- bordermode – INSIDE (the default) to indicate that other options refer to the parent's inside (ignoring the parent's border); OUTSIDE otherwise.
- height, width – Height and width in pixels.
- relheight, relwidth – Height and width as a float between 0.0 and 1.0, as a fraction of the height and width of the parent widget.
- relx, rely – Horizontal and vertical offset as a float between 0.0 and 1.0, as a fraction of the height and width of the parent widget.
- x, y – Horizontal and vertical offset in pixels.

Common Widget Properties

Common attributes such as sizes, colors and fonts are specified.

- Dimensions
- Colors
- Fonts
- Anchors
- Relief styles
- Bitmaps
- Cursors

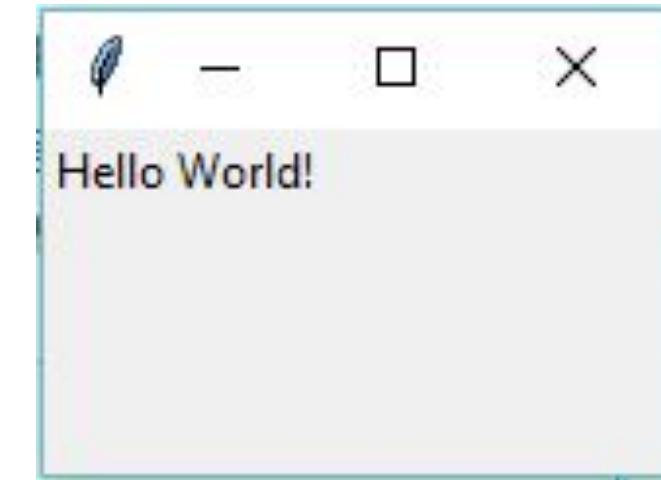
Label Widgets

- A label is a widget that displays text or images, typically that the user will just view but not otherwise interact with. Labels are used for such things as identifying controls or other parts of the user interface, providing textual feedback or results, etc.
- Syntax

tk.Label(parent, text="message")

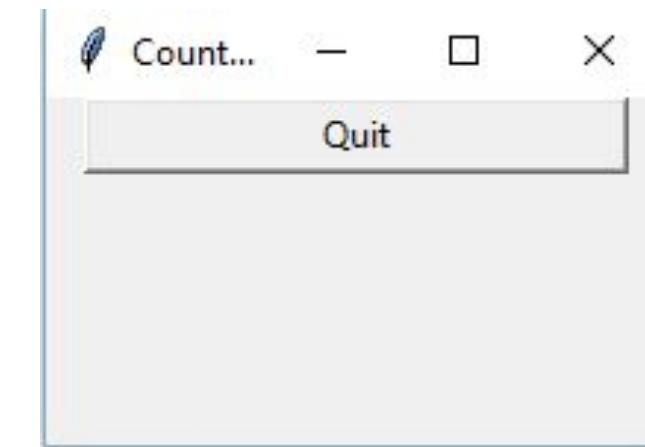
Example:

```
import tkinter as tk  
  
root = tk.Tk()  
  
label = tk.Label(root, text='Hello World!')  
  
label.grid()  
  
root.mainloop()
```



Button Widgets

```
import tkinter as tk  
  
r = tk.Tk()  
  
r.title('Counting Seconds')  
  
button = tk.Button(r, text='Stop', width=25, command=r.destroy)  
  
button.pack()  
  
r.mainloop()
```



Button Widgets

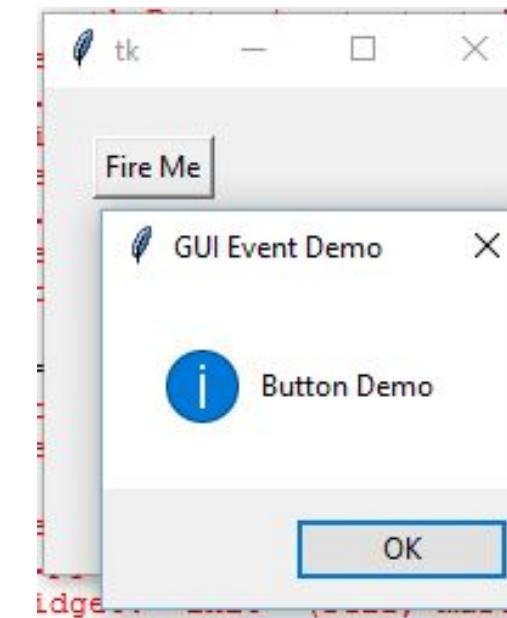
- A button, unlike a frame or label, is very much designed for the user to interact with, and in particular, press to perform some action. Like labels, they can display text or images, but also have a whole range of new options used to control their behavior.

Syntax

```
button = ttk.Button(parent, text='ClickMe', command=submitForm)
```

Example:

```
import tkinter as tk  
  
from tkinter import messagebox  
  
def hello():  
  
    msg = messagebox.showinfo( "GUI Event Demo","Button Demo")  
  
root = tk.Tk()  
  
root.geometry("200x200")  
  
b = tk.Button(root, text='Fire Me',command=hello)  
  
b.place(x=50,y=50)  
  
root.mainloop()
```



Button Widgets

- Button: To add a button in your application, this widget is used.

Syntax :

```
w=Button(master, text="caption" option=value)
```

- master is the parameter used to represent the parent window.
- activebackground: to set the background color when button is under the cursor.
- activeforeground: to set the foreground color when button is under the cursor.
- bg: to set he normal background color.
- command: to call a function.
- font: to set the font on the button label.
- image: to set the image on the button.
- width: to set the width of the button.
- height: to set the height of the button.

Entry Widgets

- An entry presents the user with a single line text field that they can use to type in a string value. These can be just about anything: their name, a city, a password, social security number, and so on.

Syntax

```
name = ttk.Entry(parent, textvariable=username)
```

Example:

```
def hello():

    msg = messagebox.showinfo( "GUI Event Demo",t.get())

root = tk.Tk()

root.geometry("200x200")

l1=tk.Label(root,text="Name:")

l1.grid(row=0)

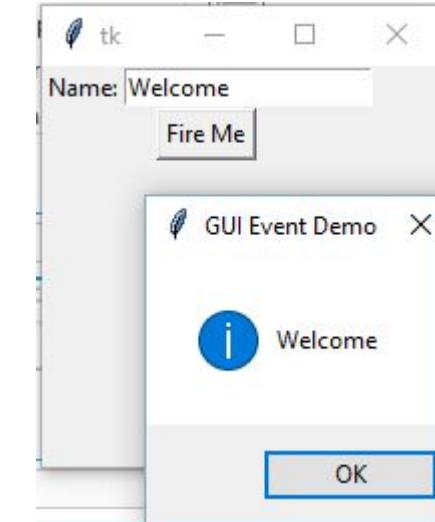
t=tk.Entry(root)

t.grid(row=0,column=1)

b = tk.Button(root, text='Fire Me',command=hello)

b.grid(row=1,columnspan=2);

root.mainloop()
```



Canvas

- The Canvas is a rectangular area intended for drawing pictures or other complex layouts. You can place graphics, text, widgets or frames on a Canvas.
- It is used to draw pictures and other complex layout like graphics, text and widgets.

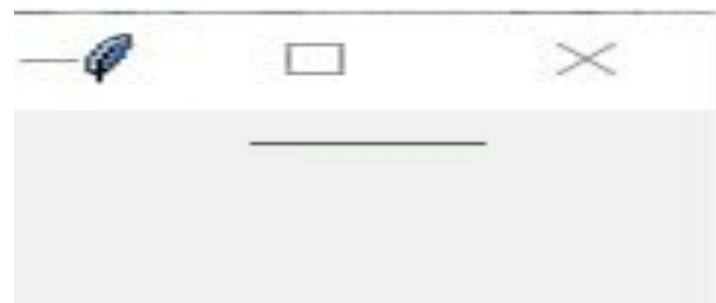
Syntax:

```
w = Canvas(master, option=value)
```

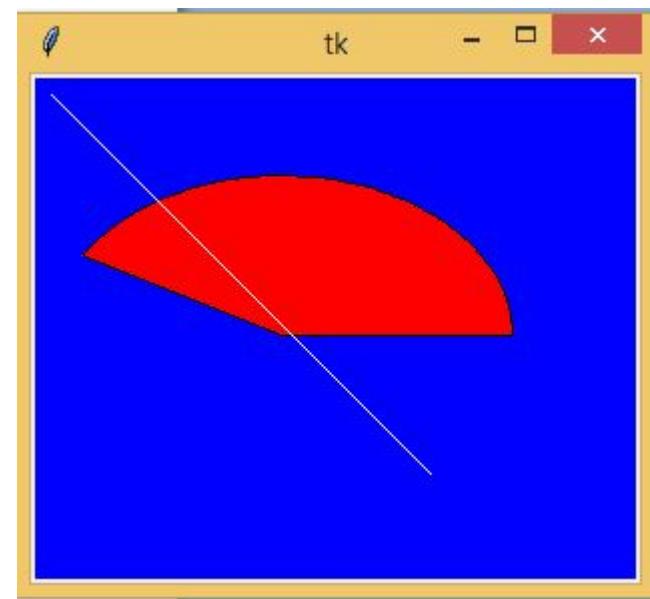
- master is the parameter used to represent the parent window.
- bd: to set the border width in pixels.
- bg: to set the normal background color.
- cursor: to set the cursor used in the canvas.
- highlightcolor: to set the color shown in the focus highlight.
- width: to set the width of the widget.
- height: to set the height of the widget.

Canvas

```
from tkinter import *\n\nmaster = Tk()\n\nw = Canvas(master, width=40, height=60)\n\nw.pack()\n\ncanvas_height=20\n\ncanvas_width=200\n\ny = int(canvas_height / 2)\n\nw.create_line(0, y, canvas_width, y )\n\nmainloop()
```



```
from tkinter import *\n\nfrom tkinter import messagebox\n\ntop = Tk()\n\nC = Canvas(top, bg = "blue", height = 250, width = 300)\n\ncoord = 10, 50, 240, 210\n\narc = C.create_arc(coord, start = 0, extent = 150, fill = "red")\n\nline = C.create_line(10,10,200,200,fill = 'white')\n\nC.pack()\n\ntop.mainloop()
```



Checkbutton

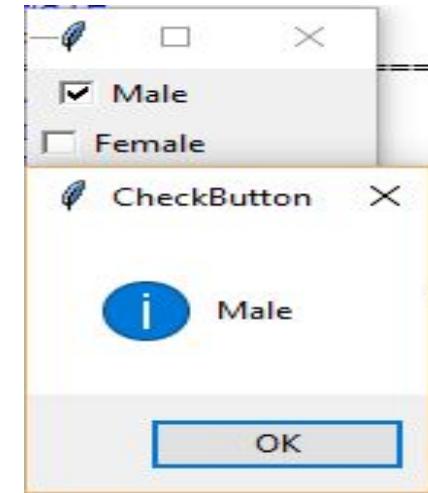
- A checkbutton is like a regular button, except that not only can the user press it, which will invoke a command callback, but it also holds a binary value of some kind (i.e. a toggle). Checkbuttons are used all the time when a user is asked to choose between, e.g. two different values for an option.

Syntax

```
w = CheckButton(master, option=value)
```

Example:

```
from tkinter import *
root= Tk()
root.title('Checkbutton Demo')
v1=IntVar()
v2=IntVar()
cb1=Checkbutton(root,text='Male', variable=v1,onvalue=1, offvalue=0, command=test)
cb1.grid(row=0)
cb2=Checkbutton(root,text='Female', variable=v2,onvalue=1, offvalue=0, command=test)
cb2.grid(row=1)
root.mainloop()
```



```
def test():
    if(v1.get()==1):
        v2.set(0)
        print("Male")
    if(v2.get()==1):
        v1.set(0)
        print("Female")
```

radiobutton

- A radiobutton lets you choose between one of a number of mutually exclusive choices; unlike a checkbutton, it is not limited to just two choices. Radiobuttons are always used together in a set and are a good option when the number of choices is fairly small

- **Syntax**

```
w = CheckButton(master, option=value)
```

Example:

```
root= Tk()  
root.geometry("200x200")  
radio=IntVar()  
rb1=Radiobutton(root,text='Red', variable=radio,width=25,value=1, command=choice)
```

```
rb1.grid(row=0)
```

```
rb2=Radiobutton(root,text='Blue', variable=radio,width=25,value=2, command=choice)
```

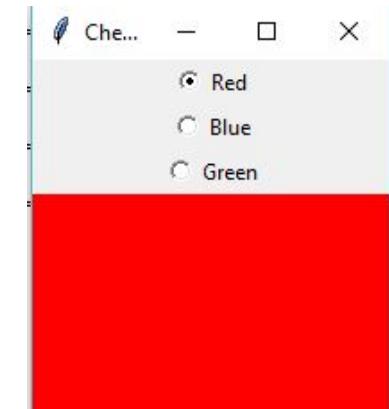
```
rb2.grid(row=1)
```

```
rb3=Radiobutton(root,text='Green', variable=radio,width=25,value=3, command=choice)
```

```
rb3.grid(row=3)
```

```
root.mainloop()
```

```
def choice():  
    if(radio.get()==1):  
        root.configure(background='red')  
    elif(radio.get()==2):  
        root.configure(background='blue')  
    elif(radio.get()==3):  
        root.configure(background='green')
```



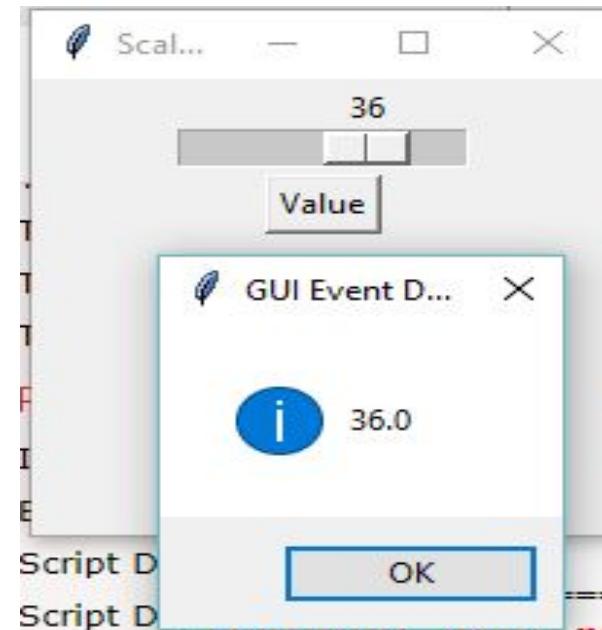
Scale

- Scale widget is used to implement the graphical slider to the python application so that the user can slide through the range of values shown on the slider and select the one among them. We can control the minimum and maximum values along with the resolution of the scale. It provides an alternative to the Entry widget when the user is forced to select only one value from the given range of values.
- **Syntax**

*w = Scale(*top, options*)*

Example:

```
from tkinter import messagebox
root= Tk()
root.title('Scale Demo')
root.geometry("200x200")
def slide():
    msg = messagebox.showinfo( "GUI Event Demo",v.get())
v = DoubleVar()
scale = Scale( root, variable = v, from_ = 1, to = 50, orient = HORIZONTAL)
scale.pack(anchor=CENTER)
btn = Button(root, text="Value", command=slide)
btn.pack(anchor=CENTER)
root.mainloop()
```



Spinbox

- The Spinbox widget is an alternative to the Entry widget. It provides the range of values to the user, out of which, the user can select the one.

Syntax

```
w = Spinbox(top, options)
```

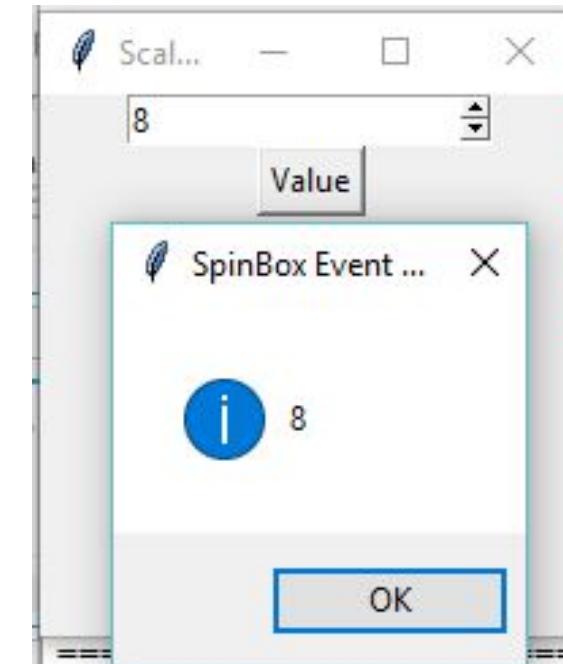
Example:

```
from tkinter import *
from tkinter import messagebox

root= Tk()
root.title('Scale Demo')
root.geometry("200x200")

def slide():
    msg = messagebox.showinfo( "SpinBox Event Demo",spin.get())

spin = Spinbox(root, from_= 0, to = 25)
spin.pack(anchor=CENTER)
btn = Button(root, text="Value", command=slide)
btn.pack(anchor=CENTER)
root.mainloop()
```



Menubutton

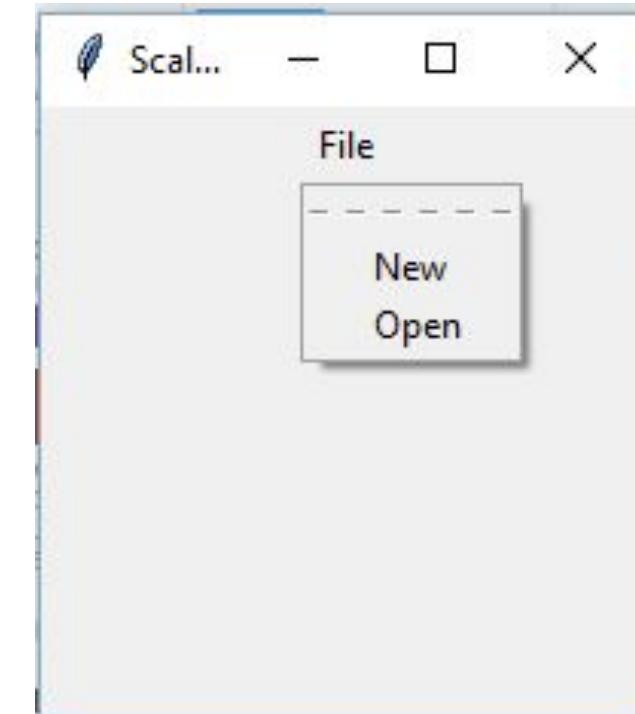
- Menubutton widget can be defined as the drop-down menu that is shown to the user all the time. It is used to provide the user a option to select the appropriate choice exist within the application.

Syntax

```
w = Menubutton(Top, options)
```

Example:

```
from tkinter import *
from tkinter import messagebox
root= Tk()
root.title('Scale Demo')
root.geometry("200x200")
menubutton = Menubutton(root, text = "File", relief = FLAT)
menubutton.grid()
menubutton.menu = Menu(menubutton)
menubutton["menu"]=menubutton.menu
menubutton.menu.add_checkbutton(label = "New", variable=IntVar(),command=)
menubutton.menu.add_checkbutton(label = "Open", variable = IntVar())
menubutton.pack()
root.mainloop()
```



Menubutton

- Menubutton widget can be defined as the drop-down menu that is shown to the user all the time. It is used to provide the user a option to select the appropriate choice exist within the application.

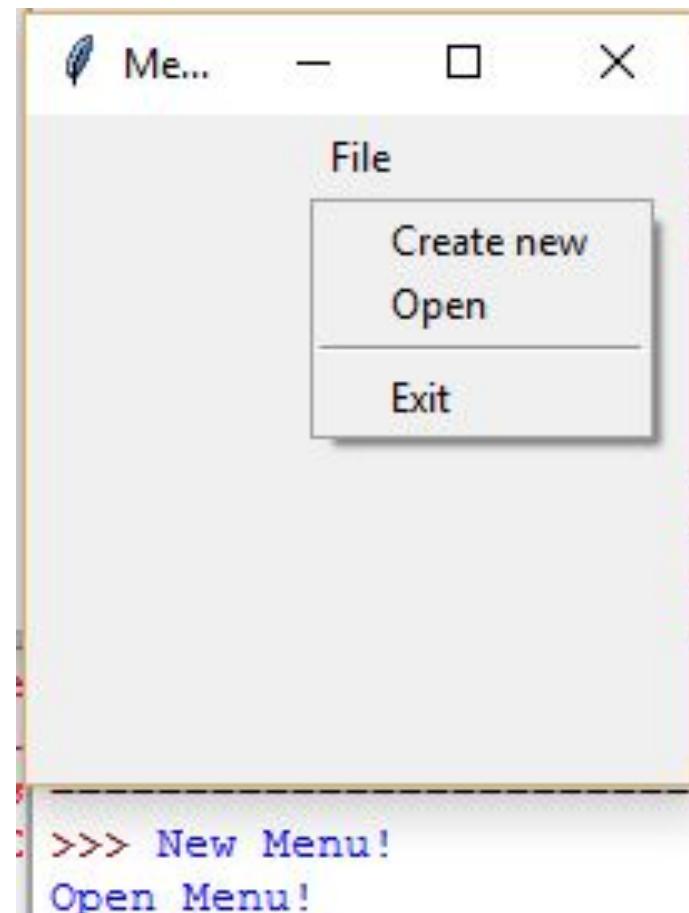
Syntax

```
w = Menubutton(Top, options)
```

Example:

```
from tkinter import *
from tkinter import messagebox
root= Tk()
root.title('Menu Demo')
root.geometry("200x200")
def new():
    print("New Menu!")
def disp():
    print("Open Menu!")

menubutton = Menubutton(root, text="File")
menubutton.grid()
menubutton.menu = Menu(menubutton, tearoff = 0)
menubutton["menu"] = menubutton.menu
menubutton.menu.add_command(label="Create
new",command=new)
menubutton.menu.add_command(label="Open",command=dis
p)
menubutton.menu.add_separator()
menubutton.menu.add_command(label="Exit",command=root.
quit)
menubutton.pack()
```

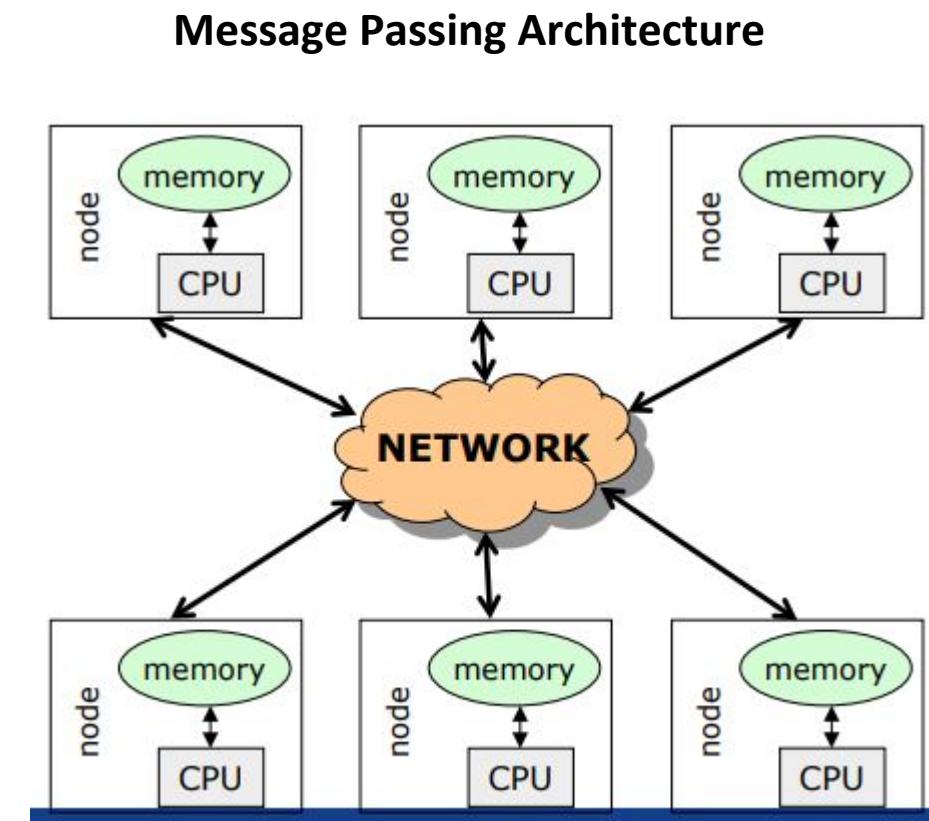
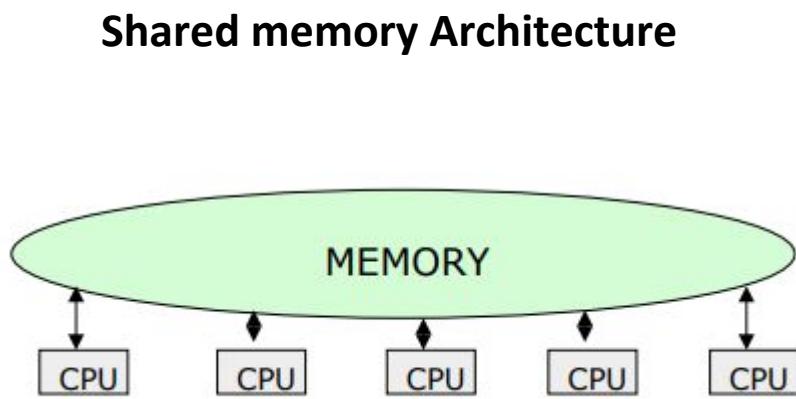


18CSC207J – Advanced Programming Practice

Parallel & Concurrent Programming Paradigm

Introduction

- A system is said to be parallel if it can support two or more actions executing simultaneously i.e., multiple actions are simultaneously executed in parallel systems.
- The evolution of parallel processing, even if slow, gave rise to a considerable variety of programming paradigms.
- Parallelism Types:
 - Explicit Parallelism
 - Implicit Parallelism



Explicit parallelism

- Explicit Parallelism is characterized by the presence of explicit constructs in the programming language, aimed at describing (to a certain degree of detail) the way in which the parallel computation will take place.
- A wide range of solutions exists within this framework. One extreme is represented by the ``ancient'' use of basic, low level mechanisms to deal with parallelism--like fork/join primitives, semaphores, etc--eventually added to existing programming languages. Although this allows the highest degree of flexibility (any form of parallel control can be implemented in terms of the basic low level primitives gif), it leaves the additional layer of complexity completely on the shoulders of the programmer, making his task extremely complicate.

Implicit Parallelism

- Allows programmers to write their programs without any concern about the exploitation of parallelism. Exploitation of parallelism is instead automatically performed by the compiler and/or the runtime system. In this way the parallelism is transparent to the programmer maintaining the complexity of software development at the same level of standard sequential programming.
- Extracting parallelism implicitly is not an easy task. For imperative programming languages, the complexity of the problem is almost prohibitively and allows positive results only for restricted sets of applications (e.g., applications which perform intensive operations on arrays).
- Declarative Programming languages, and in particular Functional and Logic languages, are characterized by a very high level of abstraction, allowing the programmer to focus on what the problem is and leaving implicit many details of how the problem should be solved.
- Declarative languages have opened new doors to automatic exploitation of parallelism. Their focusing on a high level description of the problem and their mathematical nature turned into positive properties for implicit exploitation of parallelism.

Methods for parallelism

There are many methods of programming parallel computers. Two of the most common are message passing and data parallel.

1. Message Passing - the user makes calls to libraries to explicitly share information between processors.
2. Data Parallel - data partitioning determines parallelism
3. Shared Memory - multiple processes sharing common memory space
4. Remote Memory Operation - set of processes in which a process can access the memory of another process without its participation
5. Threads - a single process having multiple (concurrent) execution paths
6. Combined Models - composed of two or more of the above.

Methods for parallelism

Message Passing:

- Each Processor has direct access only to its local memory
- Processors are connected via high-speed interconnect
- Data exchange is done via explicit processor-to-processor communication i.e processes communicate by sending and receiving messages : send/receive messages
- Data transfer requires cooperative operations to be performed by each process (a send operation must have matching receive)

Data Parallel:

- Each process works on a different part of the same data structure
- Processors have direct access to global memory and I/O through bus or fast switching network
- Each processor also has its own memory (cache)
- Data structures are shared in global address space
- Concurrent access to shared memory must be coordinate
- All message passing is done invisibly to the programmer

Steps in Parallelism

- Independently from the specific paradigm considered, in order to execute a program which exploits parallelism, the programming language must supply the means to:
 - Identify parallelism, by recognizing the components of the program execution that will be (potentially) performed by different processors;
 - Start and stop parallel executions;
 - Coordinate the parallel executions (e.g., specify and implement interactions between concurrent components).

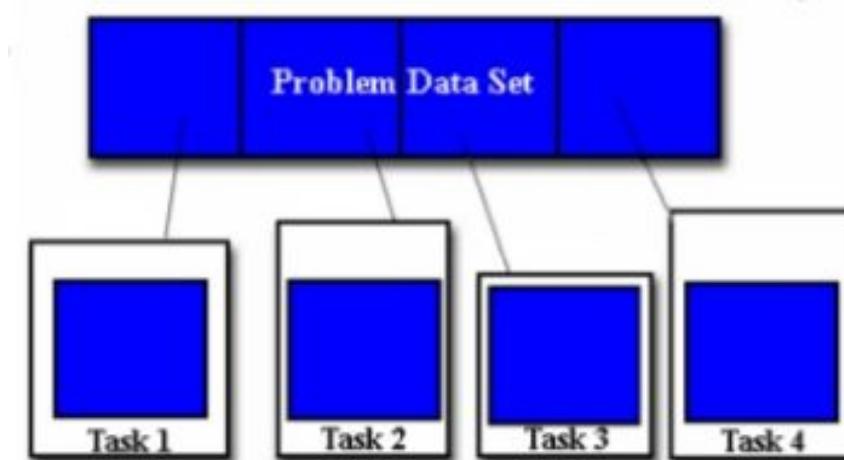
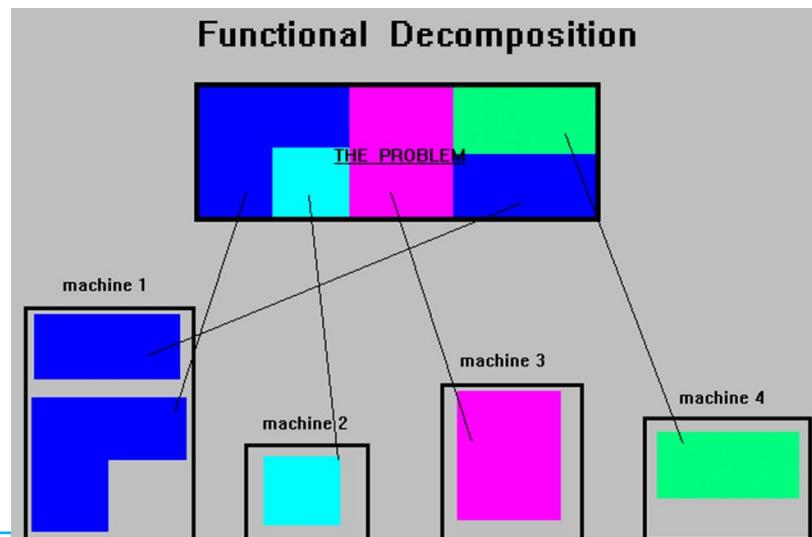
Ways for Parallelism

Functional Decomposition (Functional Parallelism)

- Decomposing the problem into different tasks which can be distributed to multiple processors for simultaneous execution
- Good to use when there is no static structure or fixed determination of number of calculations to be performed

Domain Decomposition (Data Parallelism)

- Partitioning the problem's data domain and distributing portions to multiple processors for simultaneous execution
- Good to use for problems where:
- data is static (factoring and solving large matrix or finite difference calculations)
- dynamic data structure tied to single entity where entity can be subsetted (large multi-body problems)
- domain is fixed but computation within various regions of the domain is dynamic (fluid vortices models)



Parallel Programming Paradigm

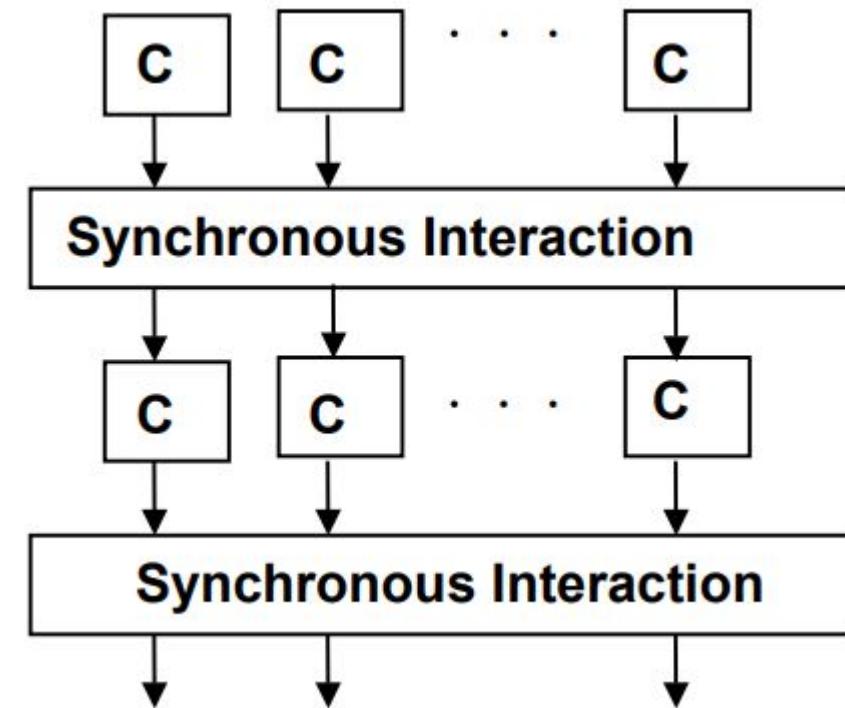
- Phase parallel
- Divide and conquer
- Pipeline
- Process farm
- Work pool

Note:

- The parallel program consists of number of super steps, and each super step has two phases : computation phase and interaction phase

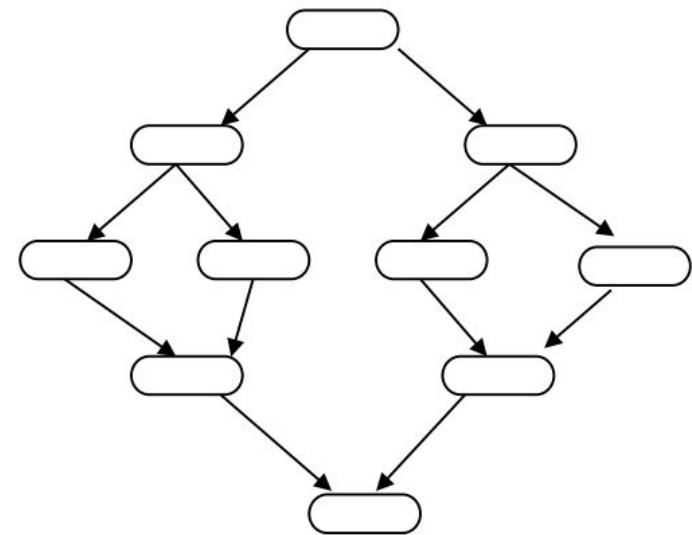
Phase Parallel Model

- The phase-parallel model offers a paradigm that is widely used in parallel programming.
- The parallel program consists of a number of supersteps, and each has two phases.
 - In a computation phase, multiple processes each perform an independent computation C.
 - In the subsequent interaction phase, the processes perform one or more synchronous interaction operations, such as a barrier or a blocking communication.
 - Then next superstep is executed.



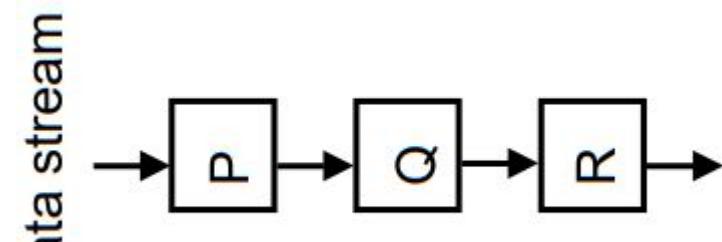
Divide and Conquer & Pipeline model

- A parent process divides its workload into several smaller pieces and assigns them to a number of child processes.
- The child processes then compute their workload in parallel and the results are merged by the parent.
- The dividing and the merging procedures are done recursively.
- This paradigm is very natural for computations such as quick sort.



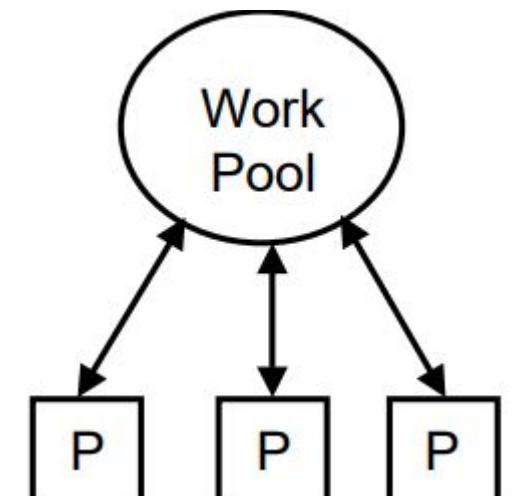
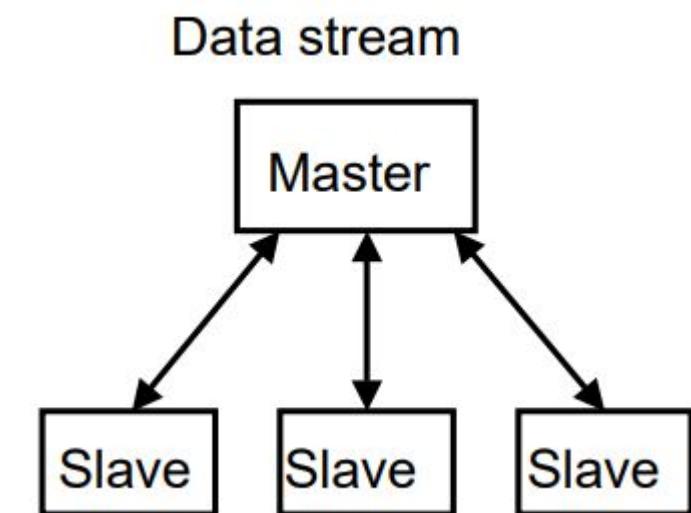
Pipeline

- In pipeline paradigm, a number of processes form a virtual pipeline.
- A continuous data stream is fed into the pipeline, and the processes execute at different pipeline stages simultaneously in an overlapped fashion.



Process Farm & Work Pool Model

- This paradigm is also known as the master-slave paradigm.
- A master process executes the essentially sequential part of the parallel program and spawns a number of slave processes to execute the parallel workload.
- When a slave finishes its workload, it informs the master which assigns a new workload to the slave.
- This is a very simple paradigm, where the coordination is done by the master.
- This paradigm is often used in a shared variable model.
- A pool of works is realized in a global data structure.
- A number of processes are created. Initially, there may be just one piece of work in the pool.
- Any free process fetches a piece of work from the pool and executes it, producing zero, one, or more new work pieces put into the pool. The parallel program ends when the work pool becomes empty.
- This paradigm facilitates load balancing, as the workload is dynamically allocated to free processes.



Parallel Program using Python

- A thread is basically an independent flow of execution. A single process can consist of multiple threads. Each thread in a program performs a particular task. For Example, when you are playing a game say FIFA on your PC, the game as a whole is a single process, but it consists of several threads responsible for playing the music, taking input from the user, running the opponent synchronously, etc.
- Threading is that it allows a user to run different parts of the program in a concurrent manner and make the design of the program simpler.
- Multithreading in Python can be achieved by importing the threading module.

Example:

```
import threading  
from threading import *
```

Parallel program using Threads in Python

```
# simplest way to use a Thread is to instantiate it with a target  
function and call start() to let it begin working.  
  
from threading import Thread,current_thread  
  
print(current_thread().getName())  
  
def mt():  
    print("Child Thread")  
  
    for i in range(11,20):  
        print(i*2)  
  
def disp():  
    for i in range(10):  
        print(i*2)  
  
child=Thread(target=mt)  
  
child.start()  
  
disp()  
  
print("Executing thread name : ",current_thread().getName())
```

```
from threading import Thread,current_thread  
  
class mythread(Thread):  
  
    def run(self):  
        for x in range(7):  
            print("Hi from child")  
  
        a = mythread()  
        a.start()  
        a.join()  
  
        print("Bye from",current_thread().getName())
```

Parallel program using Process in Python

```
import multiprocessing

def worker(num):
    print('Worker:', num)
    for i in range(num):
        print(i)
    return

jobs = []
for i in range(1,5):
    p = multiprocessing.Process(target=worker, args=(i+10,))
    jobs.append(p)
    p.start()
```

Concurrent Programming Paradigm

- Computing systems model the world, and the world contains actors that execute independently of, but communicate with, each other. In modelling the world, many (possibly) parallel executions have to be composed and coordinated, and that's where the study of concurrency comes in.
- There are two common models for concurrent programming: shared memory and message passing.
 - **Shared memory.** In the shared memory model of concurrency, concurrent modules interact by reading and writing shared objects in memory.
 - **Message passing.** In the message-passing model, concurrent modules interact by sending messages to each other through a communication channel. Modules send off messages, and incoming messages to each module are queued up for handling

Issues Concurrent Programming Paradigm

Concurrent programming is programming with multiple tasks. The major issues of concurrent programming are:

- Sharing computational resources between the tasks;
- Interaction of the tasks.

Objects shared by multiple tasks have to be safe for concurrent access. Such objects are called protected. Tasks accessing such an object interact with each other indirectly through the object.

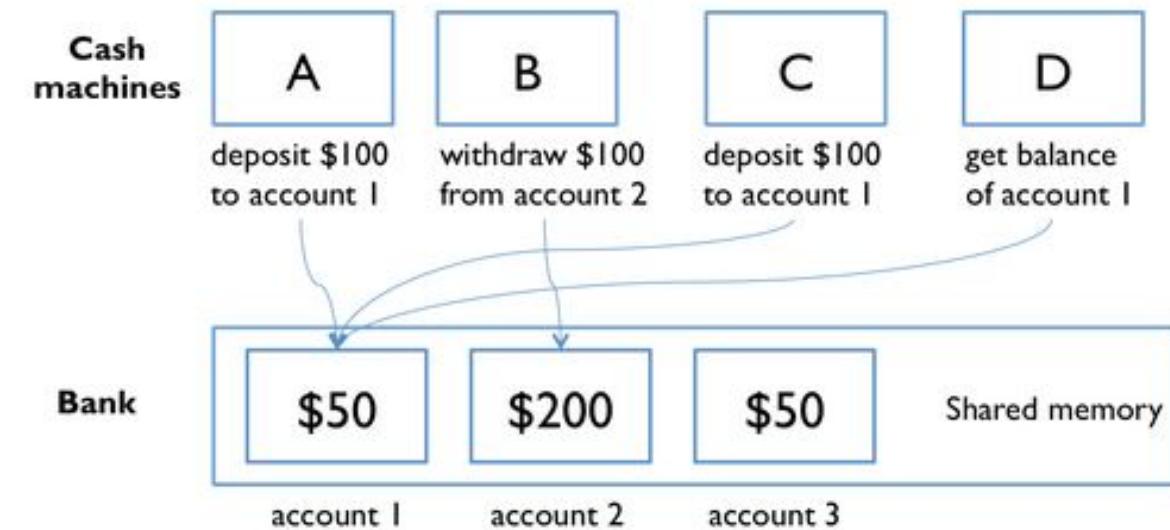
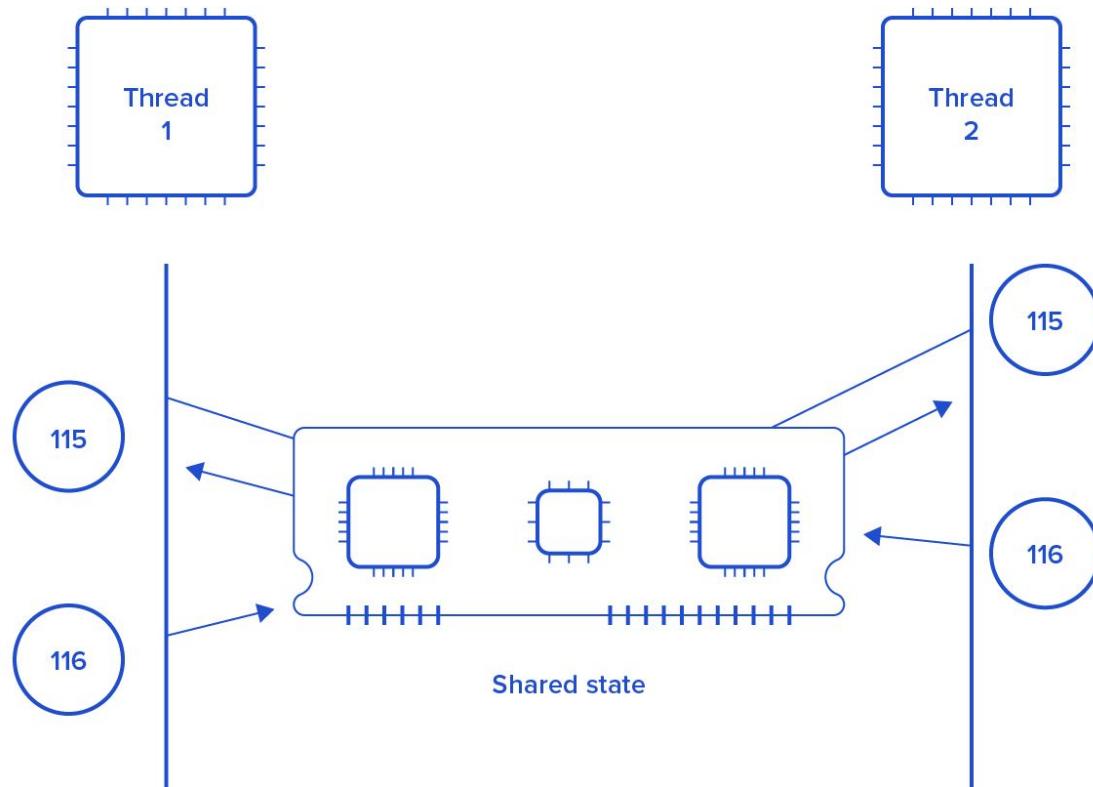
An access to the protected object can be:

- Lock-free, when the task accessing the object is not blocked for a considerable time;
- Blocking, otherwise.

Blocking objects can be used for task synchronization. To the examples of such objects belong:

- Events;
- Mutexes and semaphores;
- Waitable timers;
- Queues

Issues Concurrent Programming Paradigm



Race Condition

```
import threading
x = 0 # A shared value
COUNT = 100

def incr():
    global x
    for i in range(COUNT):
        x += 1
        print(x)

def decr():
    global x
    for i in range(COUNT):
        x -= 1
        print(x)

t1 = threading.Thread(target=incr)
t2 = threading.Thread(target=decr)
t1.start()
t2.start()
t1.join()
t2.join()
print(x)
```

Synchronization in Python

Locks:

Locks are perhaps the simplest synchronization primitives in Python. A Lock has only two states — locked and unlocked (surprise). It is created in the unlocked state and has two principal methods — acquire() and release(). The acquire() method locks the Lock and blocks execution until the release() method in some other co-routine sets it to unlocked.

R-Locks:

R-Lock class is a version of simple locking that only blocks if the lock is held by another thread. While simple locks will block if the same thread attempts to acquire the same lock twice, a re-entrant lock only blocks if another thread currently holds the lock.

Semaphore:

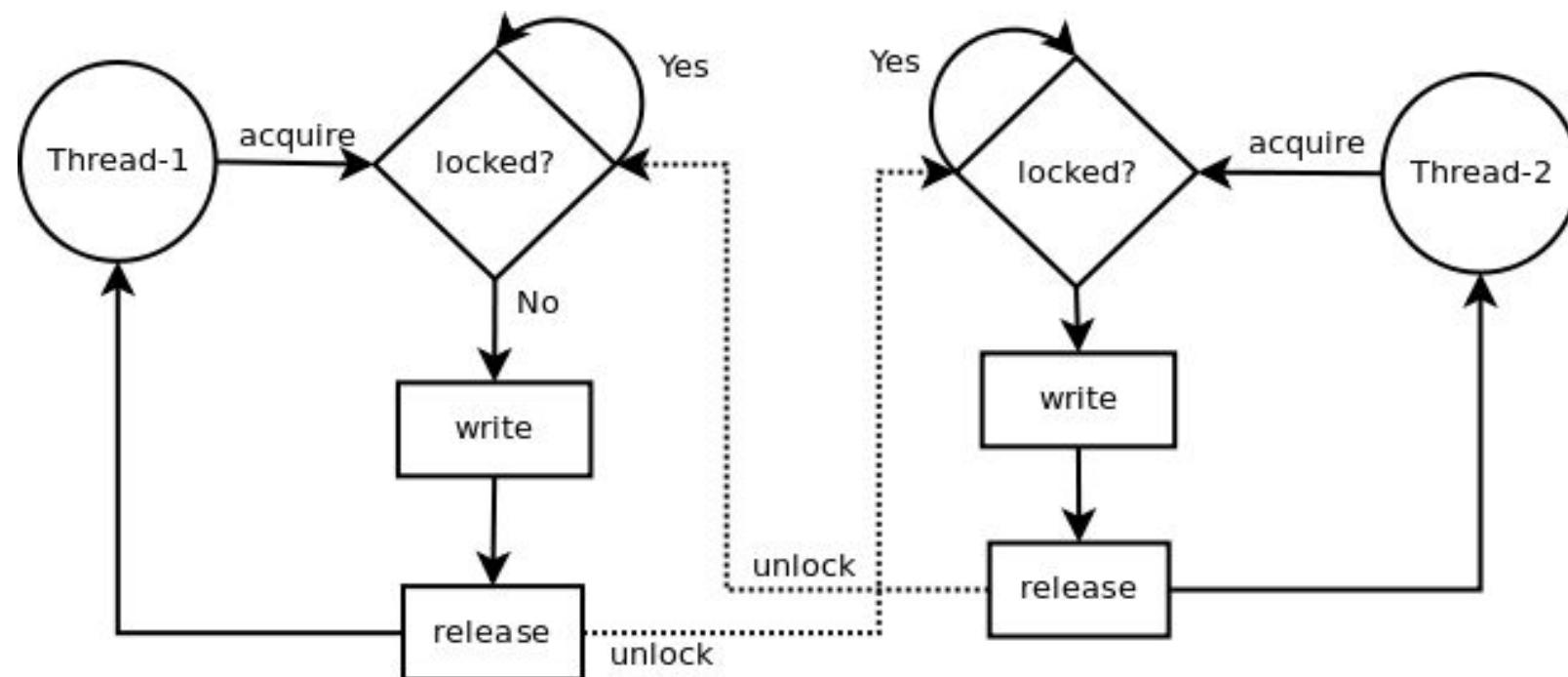
A semaphore has an internal counter rather than a lock flag, and it only blocks if more than a given number of threads have attempted to hold the semaphore. Depending on how the semaphore is initialized, this allows multiple threads to access the same code section simultaneously.

LOCK in python

Synchronization using LOCK

Locks have 2 states: locked and unlocked. 2 methods are used to manipulate them: acquire() and release(). Those are the rules:

1. if the state is unlocked: a call to acquire() changes the state to locked.
2. if the state is locked: a call to acquire() blocks until another thread calls release().
3. if the state is unlocked: a call to release() raises a RuntimeError exception.
4. if the state is locked: a call to release() changes the state to unlocked().



Synchronization in Python using Lock

```
import threading

x = 0    # A shared value
COUNT = 100
lock = threading.Lock()

def incr():
    global x
    lock.acquire()
    print("thread locked for increment cur x=",x)
    for i in range(COUNT):
        x += 1
        print(x)
    lock.release()
    print("thread release from increment cur x=",x)

def decr():
    global x
    lock.acquire()
    print("thread locked for decrement cur x=",x)
    for i in range(COUNT):
        x -= 1
        print(x)
    lock.release()
    print("thread release from decrement cur x=",x)

t1 = threading.Thread(target=incr)
t2 = threading.Thread(target=decr)
t1.start()
t2.start()
t1.join()
t2.join()
```

Synchronization in Python using RLock

```
import threading

class Foo(object):
    lock = threading.RLock()
    def __init__(self):
        self.x = 0
    def add(self,n):
        with Foo.lock:
            self.x += n
    def incr(self):
        with Foo.lock:
            self.add(1)
    def decr(self):
        with Foo.lock:
            self.add(-1)

def adder(f,count):
    while count > 0:
        f.incr()
        count -= 1

def subber(f,count):
    while count > 0:
        f.decr()
        count -= 1

# Create some threads and make sure it works
COUNT = 10
f = Foo()
t1 = threading.Thread(target=adder,args=(f,COUNT))
t2 = threading.Thread(target=subber,args=(f,COUNT))
t1.start()
t2.start()
t1.join()
t2.join()
print(f.x)
```

Synchronization in Python using Semaphore

```
import threading
import time
done = threading.Semaphore(0)
item = None
def producer():
    global item
    print "I'm the producer and I produce data."
    print "Producer is going to sleep."
    time.sleep(10)
    item = "Hello"
    print "Producer is alive. Signaling the consumer."
    done.release()
def consumer():
    print "I'm a consumer and I wait for data."
    print "Consumer is waiting."
    done.acquire()
    print "Consumer got", item
t1 = threading.Thread(target=producer)
t2 = threading.Thread(target=consumer)
t1.start()
t2.start()
```

Synchronization in Python using event

```
import threading # A producer thread
import time
item = None
# A semaphore to indicate that an item is available
available = threading.Semaphore(0)
# An event to indicate that processing is complete
completed = threading.Event()
# A worker thread
def worker():
    while True:
        available.acquire()
        print "worker: processing", item
        time.sleep(5)
        print "worker: done"
        completed.set()
def producer():
    global item
    for x in range(5):
        completed.clear() # Clear the event
        item = x # Set the item
        print "producer: produced an item"
        available.release() # Signal on the semaphore
        completed.wait()
        print "producer: item was processed"
t1 = threading.Thread(target=producer)
t1.start()
t2 = threading.Thread(target=worker)
t2.setDaemon(True)
t2.start()
```

Producer and Consumer problem using thread

```
import threading,time,Queue  
  
items = Queue.Queue()  
  
# A producer thread  
  
def producer():  
    print "I'm the producer"  
  
    for i in range(30):  
        items.put(i)  
  
        time.sleep(1)  
  
# A consumer thread  
  
def consumer():  
    print "I'm a consumer", threading.currentThread().name  
  
    while True:  
        x = items.get()  
  
        print threading.currentThread().name,"got", x  
  
        time.sleep(5)  
  
        # Launch a bunch of consumers  
        cons = [threading.Thread(target=consumer)  
                for i in range(10)]  
  
        for c in cons:  
            c.setDaemon(True)  
            c.start()  
  
        # Run the producer  
        producer()
```

Producer and Consumer problem using thread

```
import threading
import time

# A list of items that are being produced. Note: it is actually
# more efficient to use a collections.deque() object for this.

items = []
# A condition variable for items
items_cv = threading.Condition()

def producer():
    print "I'm the producer"
    for i in range(30):
        with items_cv:      # Always must acquire the lock first
            items.append(i) # Add an item to the list
            items_cv.notify() # Send a notification signal
        time.sleep(1)

def consumer():
    print "I'm a consumer", threading.currentThread().name
    while True:
        with items_cv:      # Must always acquire the lock
            while not items: # Check if there are any items
                items_cv.wait() # If not, we have to sleep
            x = items.pop(0) # Pop an item off
            print threading.currentThread().name,"got", x
            time.sleep(5)
    cons = [threading.Thread(target=consumer)
            for i in range(10)]
    for c in cons:
        c.setDaemon(True)
        c.start()
    producer()
```

Network Programming Paradigm

Introduction

The Network paradigm involves thinking of computing in terms of a client, who is essentially in need of some type of information, and a server, who has lots of information and is just waiting to hand it out. Typically, a client will connect to a server and query for certain information. The server will go off and find the information and then return it to the client.

In the context of the Internet, clients are typically run on desktop or laptop computers attached to the Internet looking for information, whereas servers are typically run on larger computers with certain types of information available for the clients to retrieve. The Web itself is made up of a bunch of computers that act as Web servers; they have vast amounts of HTML pages and related data available for people to retrieve and browse. Web clients are used by those of us who connect to the Web servers and browse through the Web pages.

Network programming uses a particular type of network communication known as sockets. A socket is a software abstraction for an input or output medium of communication.

What is Socket?

- A socket is a software abstraction for an input or output medium of communication.
- Sockets allow communication between processes that lie on the same machine, or on different machines working in diverse environment and even across different continents.
- A socket is the most vital and fundamental entity. Sockets are the end-point of a two-way communication link.
- An endpoint is a combination of IP address and the port number.

For Client-Server communication,

- Sockets are to be configured at the two ends to initiate a connection,
- Listen for incoming messages
- Send the responses at both ends
- Establishing a bidirectional communication.

Socket Types

Datagram Socket

- A datagram is an independent, self-contained piece of information sent over a network whose arrival, arrival time, and content are not guaranteed. A datagram socket uses User Datagram Protocol (UDP) to facilitate the sending of datagrams (self-contained pieces of information) in an unreliable manner. Unreliable means that information sent via datagrams isn't guaranteed to make it to its destination.

Stream Socket:

- A stream socket, or connected socket, is a socket through which data can be transmitted continuously. A stream socket is more akin to a live network, in which the communication link is continuously active. A stream socket is a "connected" socket through which data is transferred continuously.

Socket in Python

```
sock_obj = socket.socket( socket_family, socket_type, protocol=0)
```

socket_family: - Defines family of protocols used as transport mechanism.

Either AF_UNIX, or

AF_INET (IP version 4 or IPv4).

socket_type: Defines the types of communication between the two end-points.

SOCK_STREAM (for connection-oriented protocols, e.g., TCP), or

SOCK_DGRAM (for connectionless protocols e.g. UDP).

protocol: We typically leave this field or set this field to zero.

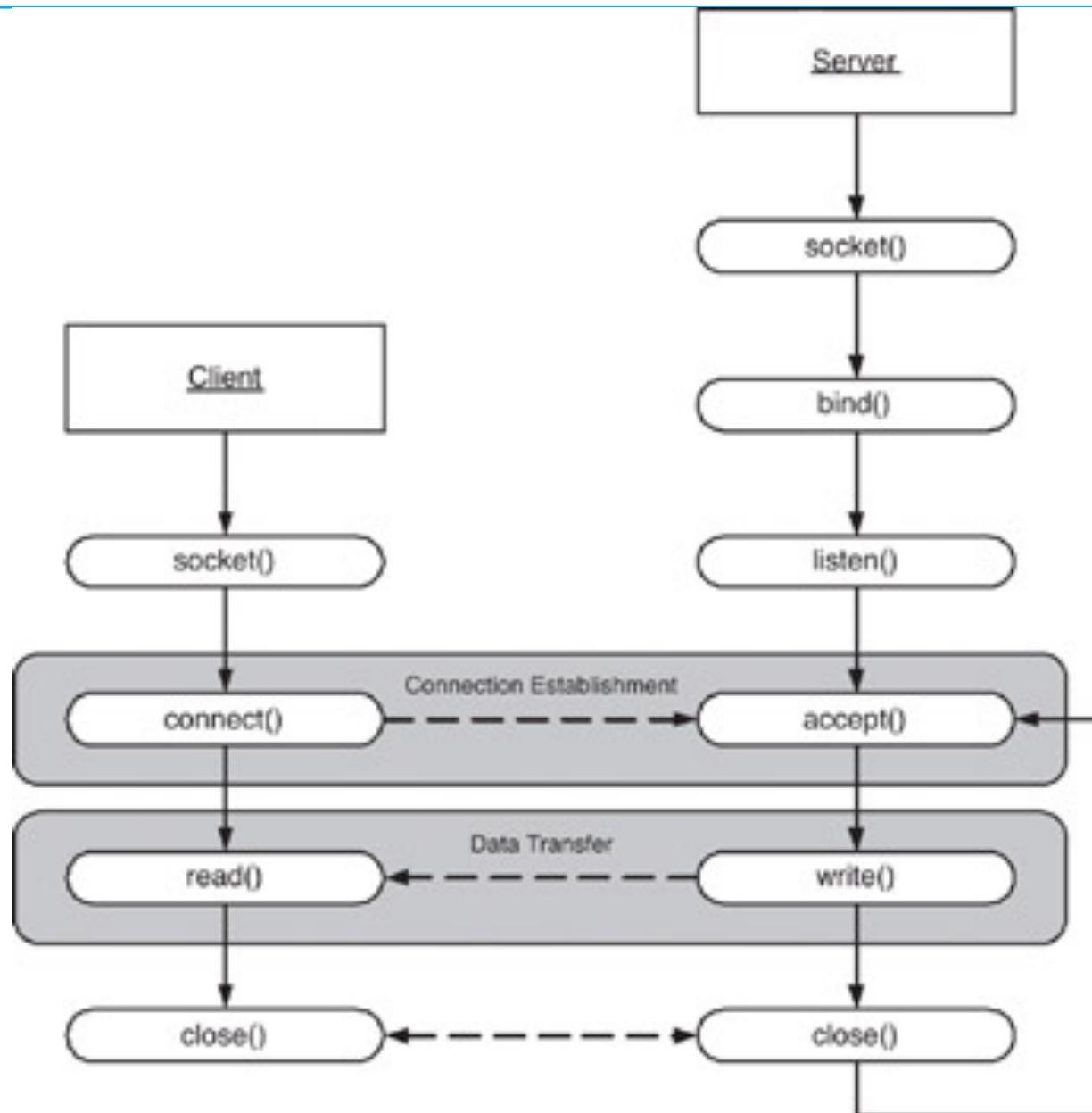
Example:

```
#Socket client example in python  
import socket  
  
#create an AF_INET, STREAM socket (TCP)  
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
print 'Socket Created'
```

Socket Creation

```
import socket  
  
import sys  
  
try:  
  
    #create an AF_INET, STREAM socket (TCP)  
  
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
  
except socket.error, msg:  
  
    print 'Failed to create socket. Error code: ' + str(msg[0]) + ' , Error message : ' + msg[1]  
  
    sys.exit();  
  
print 'Socket Created'
```

Client/server symmetry in Sockets applications



Socket in Python

To create a socket, we must use `socket.socket()` function available in the Python socket module, which has the general syntax as follows:

S = socket.socket(socket_family, socket_type, protocol=0)

`socket_family`: This is either AF_UNIX or AF_INET. We are only going to talk about INET sockets in this tutorial, as they account for at least 99% of the sockets in use.

`socket_type`: This is either SOCK_STREAM or SOCK_DGRAM.

Protocol: This is usually left out, defaulting to 0.

Client Socket Methods

Following are some client socket methods:

`connect()` : To connect to a remote socket at an address. An address format(`host, port`) pair is used for AF_INET address family.

Socket in Python

Server Socket Methods

`bind()`: This method binds the socket to an address. The format of address depends on socket family mentioned above(`AF_INET`).

`listen(backlog)` : This method listens for the connection made to the socket. The backlog is the maximum number of queued connections that must be listened before rejecting the connection.

`accept()` : This method is used to accept a connection. The socket must be bound to an address and listening for connections. The return value is a pair(`conn, address`) where `conn` is a new socket object which can be used to send and receive data on that connection, and `address` is the address bound to the socket on the other end of the connection.

General Socket in Python

`sock_object.recv():`

Use this method to receive messages at endpoints when the value of the protocol parameter is TCP.

`sock_object.send():`

Apply this method to send messages from endpoints in case the protocol is TCP.

`sock_object.recvfrom():`

Call this method to receive messages at endpoints if the protocol used is UDP.

`sock_object.sendto():`

Invoke this method to send messages from endpoints if the protocol parameter is UDP.

`sock_object.gethostname():`

This method returns hostname.

`sock_object.close():`

This method is used to close the socket. The remote endpoint will not receive data from this side.

Simple TCP Server

```
#!/usr/bin/python

#This is tcp_server.py script

import socket #line 1: Import socket module

s = socket.socket() #line 2: create a socket object
host = socket.gethostname() #line 3: Get current machine name
port = 9999 #line 4: Get port number for connection

s.bind((host,port)) #line 5: bind with the address

print "Waiting for connection..." #line 6: listen for connections

while True:
    conn,addr = s.accept() #line 7: connect and accept from client
    print 'Got Connection from', addr
    conn.send('Server Saying Hi')
    conn.close() #line 8: Close the connection
```

Simple TCP Client

```
#!/usr/bin/python

#This is tcp_client.py script

import socket

s = socket.socket()
host = socket.gethostname()          # Get current machine name
port = 9999                          # Client wants to connect to server's
                                      # port number 9999
s.connect((host,port))
print s.recv(1024)                  # 1024 is bufsize or max amount
                                      # of data to be received at once
s.close()
```

Simple UDP Server

```
#!/usr/bin/python

import socket

sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)          # For UDP

udp_host = socket.gethostname()                                # Host IP
udp_port = 12345                                              # specified port to connect

#print type(sock) =====> 'type' can be used to see type
# of any variable ('sock' here)

sock.bind((udp_host,udp_port))

while True:
    print "Waiting for client..."
    data,addr = sock.recvfrom(1024)                            #receive data from client
    print "Received Messages:",data," from",addr
```

Simple UDP Client

```
#!/usr/bin/python

import socket

sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)      # For UDP

udp_host = socket.gethostname()      # Host IP
udp_port = 12345                      # specified port to connect

msg = "Hello Python!"
print "UDP target IP:", udp_host
print "UDP target Port:", udp_port

sock.sendto(msg,(udp_host,udp_port))      # Sending message to UDP server
```

Logic + Control

What + How

**Clauses + resolution
database+interpreter**

facts+rules

If

**If X is a bird/an aeroplane, then X has wings.
Tweety is a bird**

Logic Programming Paradigm

- **Sub Topics**

- Definition - S1-SLO1
- First-class function, Higher-order function, Pure functions, Recursion- S1-SLO2
- *Packages: Kanren, SymPy – S2-SLO 1*
- *PySWIP, PyDatalog – S2-SLO 2*
- *Other languages: Prolog, ROOP, Janus S3-SLO 1*

Introduction

- It can be an abstract model of computation.
- Solve logical problems like puzzles, series
- Have **knowledge base** which we know before and along with the question
- you specify knowledge and how that knowledge is to be applied through a series of rules
- in logic programming, the common approach is to apply the methods of **resolution** and **unification**
- Knowledge is given to machine to produce result. Exp : **AL and ML code**

Introductions

- The *Logical Paradigm* takes a **declarative approach** to problem-solving.
- Various **logical assertions** about a situation are made, establishing all known facts.
- Then queries are made.
- The role of the computer becomes maintaining data and logical deduction.
- Programs are written in the language of some logic..
- Execution of a logic program is a theorem proving process; that is, computation is done by **logic inferences**
- Prolog (PROgramming in LOGic) is a) is a representative logic language
- **Exp :** Prolog, ROOP, Janus

What is a logic ?

- A logic is a language.
- It has syntax and semantics.
- More than a language, it has inference rules.
- **Syntax:**
 - The rules about how to form formulas;
 - This is usually the easy part of a logic.
- **Semantics:**
 - about the meaning carried by the formulas,
 - mainly in terms of logical consequences.
- **Inference rules**
 - Inference rules describe correct ways to derive

Features of Logical paradigms

- Computing takes place over the domain of all terms defined over a “universal” alphabet.
- Values are assigned to variables by means of automatically generated substitutions, called most general unifiers.
- These values may contain variables, called logical variables.
- The control is provided by a single mechanism: automatic backtracking.
- **Strength** - simplicity and Conciseness
- **Weakness** - has to do with the restrictions to one control mechanism and the use of a single data type.

History of Logic Programming (LP)

- Logic Programming has roots going back to early AI researchers like John McCarthy in the 50s & 60s
- Alain Colmerauer (France) designed Prolog as the first LP language in the early 1970s
- Bob Kowalski and colleagues in the UK evolved the language to its current form in the late 70s
- It's been widely used for many AI systems, but also for systems that need a fast, efficient and clean rule based engine
- The prolog model has also influenced the database community –
 - Exp datalog

General Overview

- Programs written in logic languages consist of declarations that are actually statements, or propositions, in symbolic logic.
- One of the essential characteristics of logic programming languages is their semantic
- The basic concept of this semantics is that there is a simple way to determine the meaning of each statement, and it does not depend on how the statement might be used to solve a problem.

Concepts of logic paradigm and theoretical background

- Includes both theoretical and fully implemented languages
- They all share the idea of interpreting computation as logical deduction.
- Notation Algorithm = Logic + Control
- Logic Programming uses facts and rules for solving the problem.
- That is why they are called the building blocks of Logic Programming.
- **Facts**
 - Actually, every logic program needs facts to work with so that it can achieve the given goal.
 - Facts basically are true statements about the program and data.
 - Exp : Delhi is the capital of India.
- **Rules**
 - are the constraints which allow us to make conclusions about the problem domain.
 - Basically written as logical clauses to express various facts.
 - Exp if we are building any game then all the rules must be defined.

Parts of Logical programming

1. A series of definitions/declarations that define the problem domain
2. Statements of relevant facts
3. Statement of goals in the form of a query

Advantages

- The system solves the problem, so the programming steps themselves are kept to a minimum;
- Proving the validity of a given program is simple.

Perspectives on logic programming

- Computations as Deduction
- Theorem Proving
- Non-procedural Programming
- Algorithms minus Control
- A Very High Level Programming Language
- A Procedural Interpretation of Declarative Specifications

Computation as Deduction

- Logic programming offers a slightly different paradigm for computation: computation is logical deduction
- It uses the language of logic to express data and programs.
- $\forall X, Y : X \text{ is the father of } Y \text{ if } X \text{ is a parent of } Y \text{ and } X \text{ is male}$
- Current logic programming languages use first order logic (FOL) which is often referred to as first order predicate calculus (FOPC).
- The first order refers to the constraint that we can quantify (i.e. generalize) over objects, but not over functions or relations.
- We can express "All elephants are mammals" but not
- "for every continuous function f , if $n < m$ and $f(n) < 0$ and $f(m) > 0$ then there exists an x such that $n < x < m$ and $f(x) = 0$ "

Theorem Proving

- Logic Programming uses the notion of an automatic theorem prover as an interpreter.
- The theorem prover derives a desired solution from an initial set of axioms.
- The proof must be a "constructive" one so that more than a true/false answer can be obtained
- E.G. The answer to
 - exists x such that $x = \sqrt{16}$
 - should be $x = 4$ or $x = -4$
 - rather than true

Non-procedural Programming

- Logic Programming languages are non-procedural programming languages
- A non-procedural language one in which one specifies what needs to be computed but not how it is to be done
- That is, one specifies:
 - the set of objects involved in the computation
 - the relationships which hold between them
 - the constraints which must hold for the problem to be solved
 - and leaves it up the language interpreter or compiler to decide how to satisfy the constraints

A Declarative Example

- Here's a simple way to specify what has to be true if X is the smallest number in a list of numbers L
 1. X has to be a member of the list L
 2. There can be list member X2 such that $X2 < X$

We need to say how we determine that some X is a member of a list

1. No X is a member of the empty list
2. X is a member of list L if it is equal to L's head
3. X is a member of list L if it is a member of L's tail.

Use logic to do reasoning

- Example: Given information about fatherhood and motherhood, determine grand parent relationship
- E.g. Given the information called facts
 - John is father of Lily
 - Kathy is mother of Lily
 - Lily is mother of Bill
 - Ken is father of Karen
- Who are grand parents of Bill?
- Who are grand parents of Karen?

Example

domains

being = symbol

predicates

animal(being) % all animals are beings

dog(being) % all dogs are beings

die(being) % all beings die

clauses

animal(X) :- dog(X) % all dogs are animals

dog(fido). % fido is a dog

die(X) :- animal(X) % all animals die

Logic Operators

Name	Symbol	Example	Meaning
negation	\neg	$\neg a$	not a
conjunction	\cap	$a \cap b$	a and b
disjunction	\cup	$a \cup b$	a or b
equivalence	\equiv	$a \equiv b$	a is equivalent to b
implication	\supset \subset	$a \supset b$ $a \subset b$	a implies b b implies a
universal	$\forall X.P$		For all X, P is true
existential	$\exists X.P$		There exists a value of X such that P is true

- Equivalence means that both expressions have identical truth tables
- Implication is like an if-then statement
 - if a is true then b is true
 - note that this does not necessarily mean that if a is false that b must also be false
- Universal quantifier says that this is true no matter what x is
- Existential quantifier says that there is an X that fulfills the statement

$\forall X.(\text{woman}(X) \supset \text{human}(X))$
 – if X is a woman, then X is a human

$\exists X.(\text{mother}(\text{mary}, X) \cap \text{male}(X))$
 – Mary has a son (X)

Example Statements

Consider the following knowledge:

Bob is Fred's father \square father(Bob, Fred)

Sue is Fred's mother \square mother(Sue, Fred)

Barbara is Fred's sister \square sister(Barbara, Fred)

Jerry is Bob's father \square father(Jerry, Bob)

And the following rules:

A person's father's father is the person's grandfather

A person's father or mother is that person's parent

A person's sister or brother is that person's sibling

If a person has a parent and a sibling, then the sibling has the same parent

These might be captured in first-order predicate calculus as:

$\forall x, y, z : \text{if } \text{father}(x, y) \text{ and } \text{father}(y, z) \text{ then } \text{grandfather}(x, z)$

$\forall x, y : \text{if } \text{father}(x, y) \text{ or } \text{mother}(x, y) \text{ then } \text{parent}(x, y)$

$\forall x, y : \text{if } \text{sister}(x, y) \text{ or } \text{brother}(x, y) \text{ then } \text{ sibling}(x, y) \text{ and } \text{ sibling}(y, x)$

$\forall x, y, z : \text{if } \text{parent}(x, y) \text{ and } \text{ sibling}(y, z) \text{ then } \text{parent}(x, z)$

We would rewrite these as

$\text{grandfather}(x, z) \subseteq \text{father}(x, y) \text{ and } \text{father}(y, z)$

$\text{parent}(x, y) \subseteq \text{father}(x, y)$

$\text{parent}(x, y) \subseteq \text{mother}(x, y)$ etc

Kanren

- It provides us a way to simplify the way we made code for business logic.
- It lets us express the logic in terms of rules and facts.
- The following command will help you install kanren –
 - `pip install kanren`

Matching mathematical expressions

```
from kanren import run, var, fact  
from kanren.assoccomm import eq_assoccomm as eq  
from kanren.assoccomm import commutative, associative  
add = 'add'  
mul = 'mul'  
fact(commutative, mul)  
fact(commutative, add)  
fact(associative, mul)  
fact(associative, add)  
a, b = var('a'), var('b')  
Original_pattern = (mul, (add, 5, a), b)  
exp1 = (mul, 2, (add, 3, 1))  
exp2 = (add, 5, (mul, 8, 1))  
print(run(0, (a,b), eq(original_pattern, exp1)))  
print(run(0, (a,b), eq(original_pattern, exp2)))
```

- **Output**

- ((3,2))
- ()

- **Reference :**

- [https://www.tutorialspoint.com/artificial intelligence with python/artificial intelligence with python logic programming.htm](https://www.tutorialspoint.com/artificial_intelligence_with_python/artificial_intelligence_with_python_logic_programming.htm)

SymPy Overview

Simplification

simplify

Polynomial/Rational Function Simplification

Trigonometric Simplification

Powers

Exponentials and logarithms

Special Functions

Example: Continued Fractions

Calculus

Derivatives

Integrals

Limits

Series Expansion

Finite differences

Solvers

A Note about Equations

Solving Equations Algebraically

Solving Differential Equations

Matrices

Basic Operations

Basic Methods

Matrix Constructors

Advanced Methods

Possible Issues

Reference:

- <https://www.geeksforgeeks.org/python-getting-started-with-sympy-module/>
- <https://docs.sympy.org/latest/tutorial/index.html>

SymPy module

- **Sympy** is a Python library for symbolic mathematics.
- It aims to become a full-featured computer algebra system (CAS)
- While keeping the code as simple as possible in order to be comprehensible and easily extensible.
- SymPy is written entirely in Python.
- **Installing sympy module:**
 - **Pip install sympy**

Example

```
from sympy import * x = Symbol('x')
y = Symbol('y')
z = (x + y) + (x-y)
print("value of z is :" + str(z))
```

Output:

value of z is :2*x

Find derivative, integration, limits, quadratic equation

```
# make a symbol
x = Symbol('x')
# take the derivative of sin(x)*e ^ x
ans1 = diff(sin(x)*exp(x), x)
print("derivative of sin(x)*e ^ x : ", ans1)
# Compute (e ^ x * sin(x)+ e ^ x * cos(x))dx
ans2 = integrate(exp(x)*sin(x) + exp(x)*cos(x), x)
print("indefinite integration is : ", ans2)
# Compute definite integral of sin(x ^ 2)dx
# in b / w interval of ? and ?? .
ans3 = integrate(sin(x**2), (x, -oo, oo))
print("definite integration is : ", ans3)
# Find the limit of sin(x) / x given x tends to 0
ans4 = limit(sin(x)/x, x, 0)
print("limit is : ", ans4)
# Solve quadratic equation like, example : x ^ 2?2 = 0
ans5 = solve(x**2 - 2, x)
print("roots are : ", ans5)
```

Output :

```
derivative of sin(x)*e^x : exp(x)*sin(x) + exp(x)*cos(x)
indefinite integration is : exp(x)*sin(x)
definite integration is : sqrt(2)*sqrt(pi)/2
limit is : 1
roots are : [-sqrt(2), sqrt(2)]
```

[Sympy Live Shell Demo](#)

<https://docs.sympy.org/latest/tutorial/intro.html#what-is-symbolic-computation>

Datalog Concepts

- pyDatalog is a powerful language with very few syntactic elements, mostly coming from Python :
 - Variables and expressions
 - Loops
 - Facts
 - Logic Functions and dictionaries
 - Aggregate functions
 - Literals and sets
 - Tree, graphs and recursive algorithms
 - 8-queen problem

Reference

- <https://sites.google.com/site/pydatalog/Online-datalog-tutorial>

PySwip Introduction

- PySwip is a Python - SWI-Prolog bridge enabling to query [SWI-Prolog](#) in your Python programs.
- It features an (incomplete) SWI-Prolog foreign language interface, a utility class that makes it easy querying with Prolog and also a Pythonic interface.
- Since PySwip uses SWI-Prolog as a shared library and ctypes to access it,
- it doesn't require compilation to be installed.
- Reference :
 - <https://pypi.org/project/pyswip/>

SRM Institute of Science and Technology
School of Computing

Advanced Programming Practice-18CSC207J

Paradigm types

Unit IV

- Functional Programming Paradigm [Text Book :1]
- Logic Programming Paradigm [Text Book : 1& 3]
- Dependent Type Programming Paradigm
- Network Programming Paradigm [Text Book :4]

Text Book:

1. Elad Shalom, A Review of Programming Paradigms throughout the History: With a suggestion Toward a Future Approach, Kindle Edition, 2018
2. John Goerzen, Brandon Rhodes, Foundations of Python Network Programming: The comprehensive guide to building network applications with Python, 2nd ed., Kindle Edition, 2010
3. Amit Saha, Doing Math with Python: Use Programming to Explore Algebra, Statistics,Calculus and More, Kindle Edition, 2015

Functional Programming Paradigm

Unit-III (15 Session)

Session 11-15 cover the following topics:-

- *Definition - S11-SLO1*
- *Sequence of Commands – S11-SLO2*
- *map(), reduce(), filter(), lambda – S12-SLO1*
- *partial, functools – S12-SLO2*
- *Other languages:F#, Clojure, Haskell – S13-SLO1*
- *Demo: Functional Programming in Python - S13-SLO2*

Lab 9: Functional Programming (Case Study) (S14-15)

Assignment : Comparative study of Functional programming in F#, Clojure, Haskell

- **TextBook:** Shalom, Elad. A Review of Programming Paradigms Throughout the History: With a Suggestion Toward a Future Approach, Kindle Edition

Functional Programming Paradigm (TF1)

Definition

- Mainly treat **computation to evaluate mathematical Functions**
- Avoids changing-state and mutable data
- Called as **declarative programming** paradigm
- Output depends on argument passing
- Calling same function several times with same values produces same result
- Uses expressions or declarations rather than statements as in imperative languages

Concepts

- It views all subprograms as **functions** in the mathematical sense
- Take in arguments and return a single solution.
- Solution returned is based entirely on input, and the time at which a function is called has no relevance.
- The computational model is therefore one of function application and reduction.

Characteristics of Functional Programming

- Functional programming method focuses on results, not the process
- Emphasis is on what is to be computed
- Data is immutable
- Functional programming Decompose the problem into 'functions'
- It is built on the concept of mathematical functions which uses conditional expressions and recursion to do perform the calculation
- It does not support iteration like loop statements and conditional statements like If-Else

History

- The foundation for Functional Programming is Lambda Calculus. It was developed in the 1930s for the functional application, definition, and recursion
- LISP was the first functional programming language. McCarthy designed it in 1960
- In the late 70's researchers at the University of Edinburgh defined the ML(Meta Language)
- In the early 80's Hope language adds algebraic data types for recursion and equational reasoning
- In the year 2004 Innovation of Functional language 'Scala.'

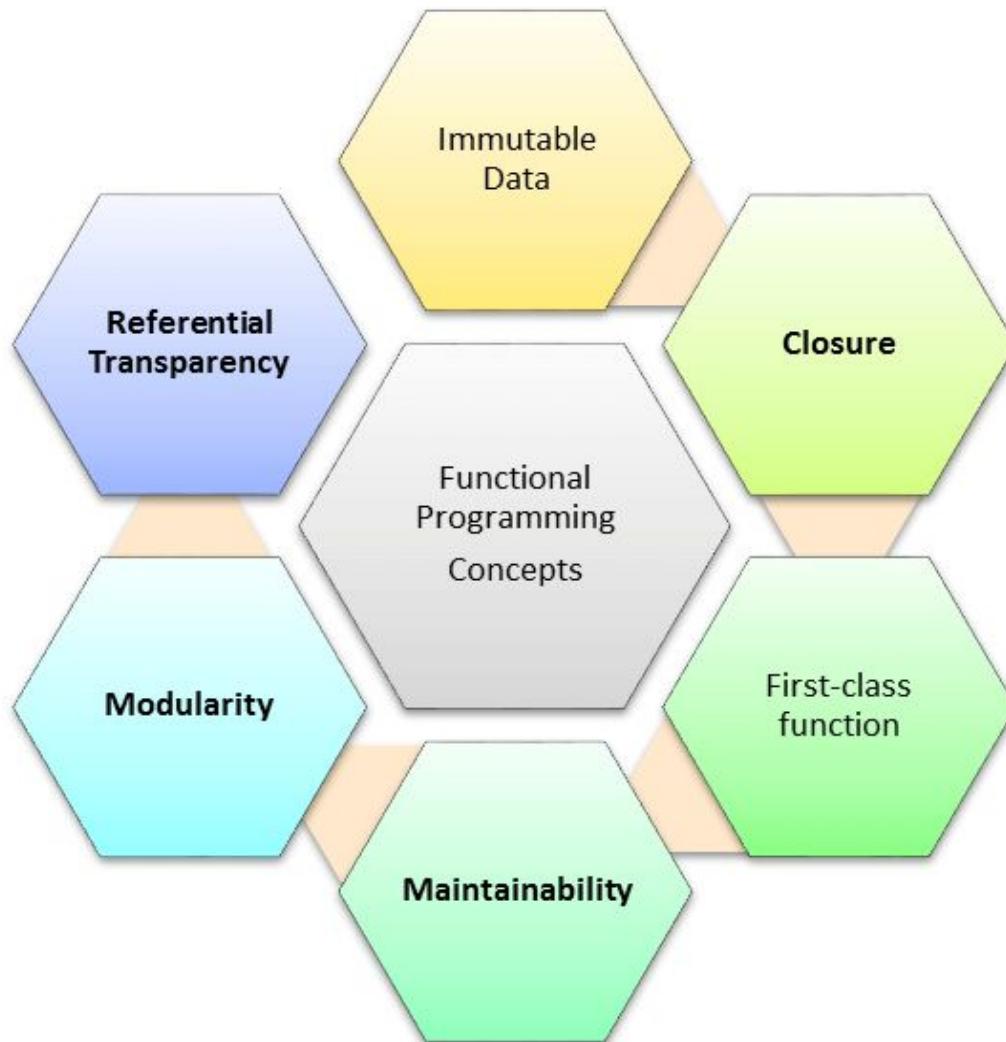
Real Time applications

- Database processing
- Financial modeling
- Statistical analysis and
- Bio-informatics

Functional Programming Languages

- Haskell
- SML
- Clojure
- Scala
- Erlang
- Clean
- F#
- ML/OCaml Lisp / Scheme
- XSLT
- SQL
- Mathematica

Basic Functional Programming Terminology and Concepts



Functional Vs Procedural

S.No	Functional Paradigms	Procedural Paradigm
1	Treats <u>computation</u> as the evaluation of <u>mathematical functions</u> avoiding <u>state</u> and <u>mutable</u> data	Derived from structured programming, based on the concept of <u>modular programming</u> or the <i>procedure call</i>
2	<u>Main traits are Lambda calculus, compositionality, formula, recursion, referential transparency</u>	Main traits are <u>Local variables</u> , sequence, selection, <u>iteration</u> , and <u>modularization</u>
3	Functional programming focuses on expressions	Procedural programming focuses on statements
4	Often recursive. Always returns the same output for a given input.	The output of a routine does not always have a direct correlation with the input.
5	Order of evaluation is usually undefined.	Everything is done in a specific order.
6	Must be stateless. i.e. No operation can have side effects.	Execution of a routine may have side effects.
7	Good fit for parallel execution, Tends to emphasize a divide and conquer approach.	Tends to emphasize implementing solutions in a linear fashion.

Functional Vs Object-oriented Programming

S.No	Functional Paradigms	Object Oriented Paradigm
1	FP uses Immutable data.	OOP uses Mutable data.
2	Follows Declarative Programming based Model.	Follows Imperative Programming Model.
3	What it focuses is on: "What you are doing. in the programme."	What it focuses is on "How you are doing your programming."
4	Supports Parallel Programming.	No supports for Parallel Programming.
5	Its functions have no-side effects.	Method can produce many side effects.
6	Flow Control is performed using function calls & function calls with recursion.	Flow control process is conducted using loops and conditional statements.
7	Execution order of statements is not very important.	Execution order of statements is important.
8	Supports both "Abstraction over Data" and "Abstraction over Behavior."	Supports only "Abstraction over Data".

Example

Functional Programming

```
num = 1  
def function_to_add_one(num):  
    num += 1  
    return num  
  
function_to_add_one(num)  
function_to_add_one(num)  
function_to_add_one(num)  
function_to_add_one(num)  
function_to_add_one(num)
```

#Final Output: 2

Procedural Programming

```
num = 1  
def procedure_to_add_one():  
    global num  
    num += 1  
    return num  
  
procedure_to_add_one()  
procedure_to_add_one()  
procedure_to_add_one()  
procedure_to_add_one()  
procedure_to_add_one()
```

#Final Output: 6

Features of Functional paradigms

- **First-class functions** – accept another function as an argument or return a function
- **Pure functions** - they are functions without side effects
- **Recursion** - allows writing smaller algorithms and operating by looking only at the inputs to a function
- **Immutable variables** - variables that cannot be changed
- **Non-strict evaluation** - allows having variables that have not yet been computed
- **Statements** - evaluable pieces of code that have a return value
- **Pattern matching** - allows better type-checking and extracting elements from an object

First-class function

- First-class functions are functions treated as objects themselves,
- Meaning they can be passed as a parameter to another function, returned as a result of a function
- It is said that a programming language has first-class functions if it treats functions as first-class citizens.

Properties of First-Class Functions

- It is an instance of an Object type
- Functions can be stored as variable
- Pass First Class Function as argument of some other functions
- Return Functions from other function
- Store Functions in lists, sets or some other data structures.

Functions as first-class objects in python

- Using functions as first-class objects means to use them in the same manner that you use data.
- So, you can pass them as parameters like passing a function to another function as an argument.
- `>>> list(map(int, ["1", "2", "3"])) [1, 2, 3]`

Higher-order function

- takes one or more functions as an input
- outputs a function
- Exp : Map Function
- Consider a function that prints a line multiple times:
 - Example Code

```
def write_repeat(message, n):  
    for i in range(n):  
        print(message)  
    write_repeat('Hello', 5)
```

Pure Functions

- **is idempotent** — returns the same result if provided the same arguments,
- has no side effects.
- If a function uses an object from a higher scope or random numbers, communicates with files and so on, it might be *impure*
- Since its result doesn't depend only on its arguments.

Example

```
def multiply_2_pure(numbers):
    new_numbers = []
    for n in numbers:
        new_numbers.append(n * 2)
    return new_numbers
```

```
original_numbers = [1, 3, 5, 10]
changed_numbers =
    multiply_2_pure(original_numbers)
print(original_numbers) # [1, 3, 5, 10]
print(changed_numbers) # [2, 6, 10, 20]
```

Anonymous Functions

- Anonymous (lambda) functions can be very convenient for functional programming constructs.
- They don't have names and usually are created ad-hoc, with a single purpose.
- Exp1: `lambda x, y: x + y`
- Exp 2: `>>> f = lambda x, y: x + y`
`>>> def g(x, y):`
`return x + y`

Examples

Anonymous function assigned to a variable. Easy to pass around and invoke when needed.

```
const myVar = function(){console.log('Anonymous function here!')}  
myVar()
```

Anonymous function as argument

- setInterval(function(){console.log(new Date().getTime())}, 1000);

Anonymous functions within a higher order function

```
function mealCall(meal){  
    return function(message){  
        return console.log(message + " " + meal + '!')  
    }  
}
```

```
const announceDinner = mealCall('dinner')
```

```
const announceLunch = mealCall('breakfast')
```

```
announceDinner('hey!, come and get your')
```

```
announceLunch('Rise and shine! time for')
```

Example using Python

- Anonymous functions are split into two types: lambda functions and closures.
- Functions are made of **four parts**: name, parameter list, body, and return

Example :

A = 5

```
def impure_sum(b):
    return b + A
def pure_sum(a, b):
    return a + b
print(impure_sum(6))
>> 11
print(pure_sum(4, 6))
>> 10
```

Example Code

Function for computing the average of two numbers:

```
(defun avg(X Y) (/ (+ X Y)  
2.0))
```

Function is called by:

```
> (avg 10.0 20.0)
```

Function returns:

```
15.0
```

Functional style of getting a sum of a list:

```
new_lst = [1, 2, 3, 4]  
def sum_list(lst):  
    if len(lst) == 1:  
        return lst[0]  
    else:  
        return lst[0] + sum_list(lst[1:])  
print(sum_list(new_lst))
```

or the pure functional way in python using higher order function

```
import functools  
print(functools.reduce(lambda x, y: x + y, new_lst))
```

Immutable variables

- Immutable variable (object) is a variable whose state cannot be modified once it is created.
- In contrast, a mutable variable can be modified after it is created
- **Exp**
- String str = “A Simple String.”;
- str.toLowerCase();

Other Examples

- int i = 12; //int is a primitive type
- i = 30; //this is ok
- final int j = 12;
- j = 30;
- final MyClass c = new MyClass();
- m.field = 100; //this is ok;
- m = new MyClass();

Functional programming tools

- **filter(*function, sequence*)**

```
def f(x): return x%2 != 0 and x%3 ==0  
filter(f, range(2,25))
```

- **map(*function, sequence*)**

- call function for each item
- return list of return values

- **reduce(*function, sequence*)**

- return a single value
- call binary function on the first two items
- then on the result and next item
- iterate

Lambda Expression

```
# Using `def` (old way).
```

```
def old_add(a, b):  
    return a + b
```

```
# Using `lambda` (new way).
```

```
new_add = lambda a, b: a + b  
old_add(10, 5) == new_add(10, 5)  
>> True
```

```
unsorted = [('b', 6), ('a', 10), ('d', 0), ('c', 4)]
```

```
print(sorted(unsorted, key=lambda x: x[1]))
```

```
>> [('d', 0), ('c', 4), ('b', 6), ('a', 10)]
```

Map Function

- Takes in an iterable (ie. list), and creates a new iterable object, a special map object.
- The new object has the first-class function applied to every element.

Pseudocode for map.

```
def map(func, seq):
    return Map(
        func(x)
        for x in seq
    )
```

Example:

```
values = [1, 2, 3, 4, 5]
```

```
add_10 = list(map(lambda x: x + 10, values))
add_20 = list(map(lambda x: x + 20, values))
```

```
print(add_10)
>> [11, 12, 13, 14, 15]
```

```
print(add_20)
>> [21, 22, 23, 24, 25]
```

Filter Function

- Takes in an iterable (ie. list), and creates a new iterable object
- The new object has the first-class function applied to every element.

Pseudocode for map.

```
return Map(  
    x for x in seq  
    if evaluate(x) is True  
)
```

Example:

```
even = list(filter(lambda x: x % 2 == 0,  
                  values))  
odd = list(filter(lambda x: x % 2 == 1, values))
```

```
print(even)  
>> [2, 4, 6, 8, 10]
```

```
print(odd)  
>> [1, 3, 5, 7, 9]
```

Advantages

- Allows you to avoid confusing problems and errors in the code
- Easier to test and execute Unit testing and debug FP Code.
- Parallel processing and concurrency
- Hot code deployment and fault tolerance
- Offers better modularity with a shorter code
- Increased productivity of the developer
- Supports Nested Functions
- Functional Constructs like Lazy Map & Lists, etc.
- Allows effective use of Lambda Calculus

Disadvantages

- Perhaps less efficiency
- Problems involving many variables, or a lot of sequential activity are sometimes easier to handle imperatively
- Functional programming paradigm is not easy, so it is difficult to understand for the beginner
- Hard to maintain as many objects evolve during the coding
- Needs lots of mocking and extensive environmental setup
- Re-use is very complicated and needs constantly refactoring
- Objects may not represent the problem correctly

Transforming code from imperative to functional

1. Introduce higher-order functions.
2. Convert existing methods into pure functions.
3. Convert loops over to recursive/tail-recursive methods (if possible).
4. Convert mutable variables into immutable variables.
5. Use pattern matching (if possible).

Mathematical Background

- For example, if $f(x)=x^2$ and x is 3, the ordered pair is $(3, 9)$.
- A function is defined by its set of inputs, called the domain.
- A set containing the set of outputs, and possibly additional elements as members, is called its codomain.
- The set of all input-output pairs is called its graph.

Mathematical Background

General Concepts

- **Notation**

$$F(x) = y$$

x , y -> Arguments or
parameters

x -> domain and
y->codomain

Types:

- Injective if $f(a) \neq f(b)$
- Surjective if $f(X) = Y$
- Bijective (support both)

Functional Rules

1. $(f+g)(x)=f(x)+g(x)$
2. $(f-g)(x)=f(x)-g(x)$
3. $(f*g)(x)=f(x)*g(x)$
4. $(f/g)(x)=f(x)/g(x)$
5. $(gof)(x)=g(f(x))$
6. $f f^{-1}=\text{id}_y$

Scenario1

- What if we wanted to write to a file 5 times, or log the message 5 times? Instead of writing 3 different functions that all loop, we can write 1 Higher Order Function that accepts those functions as an argument:

```
def hof_write_repeat(message, n, action):  
    for i in range(n):  
        action(message)
```

```
hof_write_repeat('Hello', 5, print)
```

```
# Import the logging library  
import logging  
  
# Log the output as an error instead  
hof_write_repeat('Hello', 5, logging.error)
```

Scenario2

- Imagine that we're tasked with creating functions that increment numbers in a list by 2, 5, and 10. So instead of creating many different increment functions, we create 1 Higher Order Function:

```
def hof_add(increment):  
    # Create a function that loops and adds  
    # the increment  
    def add_increment(numbers):  
        new_numbers = []  
        for n in numbers:  
            new_numbers.append(n +  
                increment)  
        return new_numbers  
    # We return the function as we do any  
    # other value  
    return add_increment  
add2=hof_add(2)  
print(add2([23, 88])) # [25, 90]  
add5 = hof_add(5)  
print(add5([23, 88])) # [28, 93]  
add10 = hof_add(10)  
print(add10([23, 88])) # [33, 98]
```

Sample Scenario

1) (The MyTriangle module) Create a module named MyTriangle that contains the following two functions:

Returns true if the sum of any two sides is greater than the third side.

def isValid(side1, side2, side3):

Returns the area of the triangle.

def area(side1, side2, side3):

Write a test program that reads three sides for a triangle and computes the area if the input is valid. Otherwise, it displays that the input is invalid.

2) A prime number is called a Mersenne prime if it can be written in the form for some positive integer p. Write a program that finds all Mersenne primes with and displays the output as follows:

p $2^p - 1$

2 3

3 7

5 31

Sample Scenarios

- 3) Write a function named `ack` that evaluates the Ackermann function. Use your function to evaluate `ack(3, 4)`, which should be 125.

The Ackermann function, $A(m, n)$, is defined:

$$A(m, n) = \begin{cases} n + 1 & \text{if } m = 0 \\ A(m - 1, 1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m - 1, A(m, n - 1)) & \text{if } m > 0 \text{ and } n > 0. \end{cases}$$

Thank you

Summary

Symbolic Programming Paradigm

Introduction

Symbolic computation deals with the computation of mathematical objects symbolically. This means that the mathematical objects are represented exactly, not approximately, and mathematical expressions with unevaluated variables are left in symbolic form.

It Covers the following:

- As A calculator and symbols
- Algebraic Manipulations - Expand and Simplify
- Calculus – Limits, Differentiation, Series , Integration
- Equation Solving – Matrices

Calculator and Symbols

Rational - $\frac{1}{2}$, or 5/2

```
>>import sympy as sym  
>>a = sym.Rational(1, 2)  
>>a
```

Answer will be $\frac{1}{2}$

Constants like pi,e

```
>>sym.pi**2  
>>sym.pi.evalf()  
>>(sym.pi + sym.exp(1)).evalf()
```

Answer is π^{**2}
Answer is 3.14159265358979

Answer is 5.85987448204884

X AND Y

```
>>x = sym.Symbol('x')  
>>y = sym.Symbol('y')  
>>x + y + x - y
```

Answer is $2*x$

Algebraic Manipulations

EXPAND (X+Y)3 = X+3X^2Y+3XY^2+Y**

>> sym.expand((x + y) ** 3) Answer is $x^{**3} + 3*x^{**2}*y + 3*x*y^{**2} + y^{**3}$

>> 3 * x * y ** 2 + 3 * y * x ** 2 + x ** 3 + y ** 3 Answer is $x^{**3} + 3*x^{**2}*y + 3*x*y^{**2} + y^{**3}$

WITH TRIGNOMETRY LIKE SIN,COSINE

eg. $\cos(X+Y) = -\sin(X)\sin(Y)+\cos(X)\cos(Y)$

>> sym.expand(sym.cos(x + y), trig=True) Answer is $-\sin(x)\sin(y) + \cos(x)\cos(y)$

SIMPLIFY

$$(X+X*Y/X)=Y+1$$

>> sym.simplify((x + x * y) / x) Answer is: $y+1$

Calculus

LIMITS compute the limit of

`limit(function, variable, point)`

`limit(sin(x)/x , x, 0) =1`

Differentiation

`diff(func,var) eg diff(sin(x),x)=cos(x)`

`diff(func,var,n) eg`

Series

`series(expr,var)`

`series(cos(x),x) = 1-x/2+x/24+o(x)`

Integration

`Integrate(expr,var)`

`Integrate(sin(x),x) = -cos(x)`

Example

Example:

```
In [23]: sym.expand(sym.cos(x + y), trig=True)
```

```
Out[23]: -sin(x)*sin(y) + cos(x)*cos(y)
```

```
In [24]: sym.limit(sym.sin(x) / x, x, 0)
```

```
Out[24]: 1
```

```
In [26]: sym.diff(sym.sin(x), x)
```

```
Out[26]: cos(x)
```

```
In [27]: sym.diff(sym.sin(2 * x), x)
```

```
Out[27]: 2*cos(2*x)
```

```
In [28]: sym.diff(sym.tan(x), x)
```

```
Out[28]: tan(x)**2 + 1
```

```
In [29]: sym.diff(sym.sin(2 * x), x, 1)
```

```
Out[29]: 2*cos(2*x)
```

```
In [30]: sym.diff(sym.sin(2 * x), x, 2)
```

```
Out[30]: -4*sin(2*x)
```

```
In [31]: sym.diff(sym.sin(2 * x), x, 3)
```

```
Out[31]: -8*cos(2*x)
```

```
In [31]: sym.diff(sym.sin(2 * x), x, 3)
```

```
Out[31]: -8*cos(2*x)
```

```
In [32]: sym.series(sym.cos(x), x)
```

```
Out[32]: 1 - x**2/2 + x**4/24 + O(x**6)
```

```
In [34]: sym.integrate(6 * x ** 5, x)
```

```
Out[34]: x**6
```

```
In [35]: sym.integrate(sym.sin(x), x)
```

```
Out[35]: -cos(x)
```

```
In [36]: sym.integrate(sym.log(x), x)
```

```
Out[36]: x*log(x) - x
```

```
In [37]: sym.integrate(2 * x + sym.sinh(x), x)
```

```
Out[37]: x**2 + cosh(x)
```

```
In [37]: sym.integrate(2 * x + sym.sinh(x), x)
```

```
Out[37]: x**2 + cosh(x)
```

```
In [38]: sym.integrate(sym.exp(-x ** 2) * sym.erf(x), x)
```

```
Out[38]: sqrt(pi)*erf(x)**2/4
```

```
In [39]: sym.integrate(x**3, (x, -1, 1))
```

```
Out[39]: 0
```

```
In [40]: sym.integrate(sym.sin(x), (x, 0, sym.pi / 2))
```

```
Out[40]: 1
```

Example

Example:

```
In [43]: sym.solveset(x ** 4 - 1, x)
```

```
Out[43]: {-1, 1, -I, I}
```

```
In [44]: sym.solveset(sym.exp(x) + 1, x)
```

```
Out[44]: ImageSet(Lambda(_n, I*(2*_n*pi + pi)), S.Integers)
```

```
In [46]: solution = sym.solve((x + 5 * y - 2, -3 * x + 6 * y - 15), (x, y))  
solution[x], solution[y]
```

```
Out[46]: (-3, 1)
```

```
In [47]: f = x ** 4 - 3 * x ** 2 + 1  
sym.factor(f)
```

```
Out[47]: (x**2 - x - 1)*(x**2 + x - 1)
```

```
In [48]: sym.satisfiable(x & y)
```

```
Out[48]: {x: True, y: True}
```

```
In [49]: sym.Matrix([[1, 0], [0, 1]])
```

```
Out[49]: Matrix([  
    [1, 0],  
    [0, 1]])
```

```
In [51]: x, y = sym.symbols('x, y')  
A = sym.Matrix([[1, x], [y, 1]])  
A
```

```
Out[51]: Matrix([  
    [1, x],  
    [y, 1]])
```

```
In [52]: A**2
```

```
Out[52]: Matrix([  
    [x*y + 1, 2*x],  
    [2*y, x*y + 1]])
```

Equation Solving

solveset()

solveset($x^{**} 4 - 1$, x) ={-1,1,-I,I}

Matrices

A={[1,2][2,1]} find A**2

Automata Based Programming Paradigm

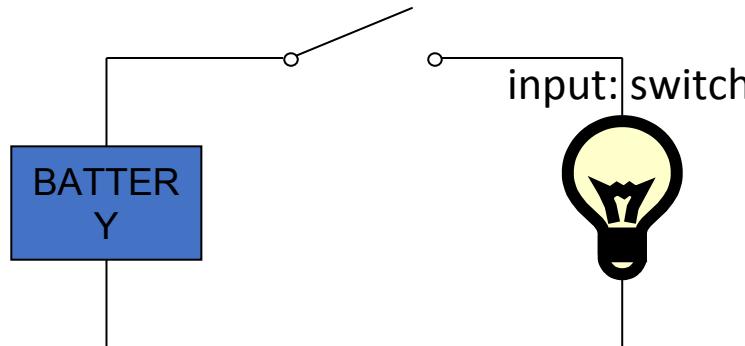
Introduction

Automata-based programming is a programming paradigm in which the program or its part is thought of as a model of a finite state machine or any other formal automation.

What is Automata Theory?

- Automata theory is the study of abstract computational devices
- Abstract devices are (simplified) models of real computations
- Computations happen everywhere: On your laptop, on your cell phone, in nature, ...

Example:



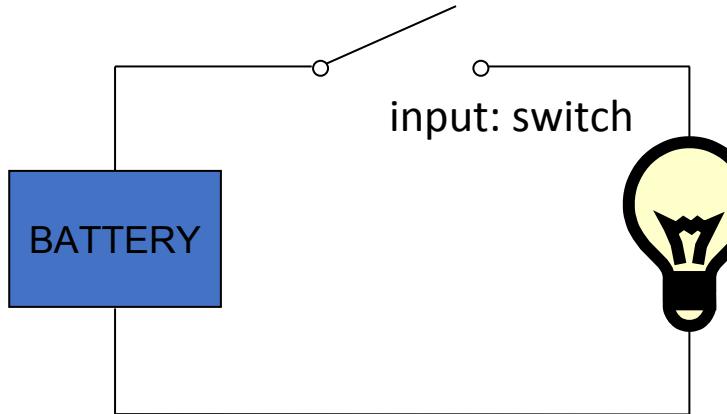
output: light bulb

actions: flip switch

states: on, off

Simple Computer

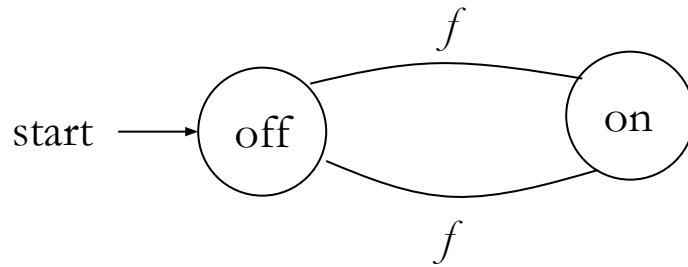
Example:



output: light bulb

actions: flip switch

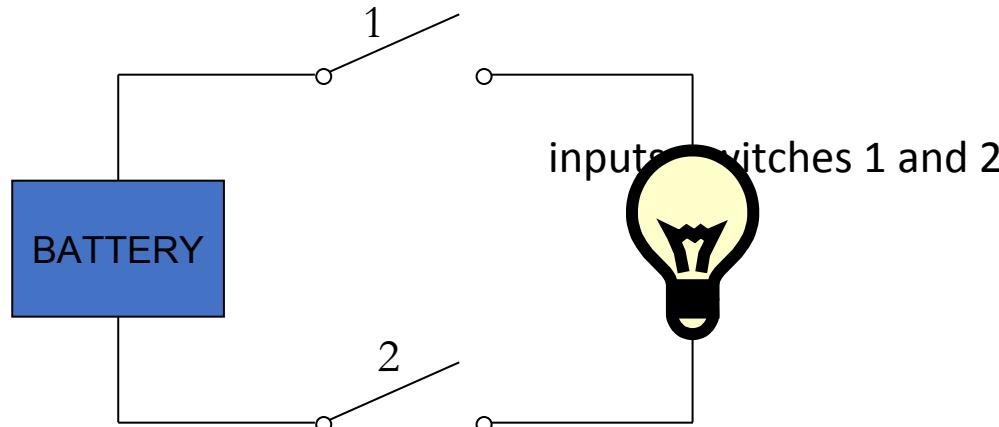
states: on, off



bulb is on if and only if there was an odd number of flips

Another “computer”

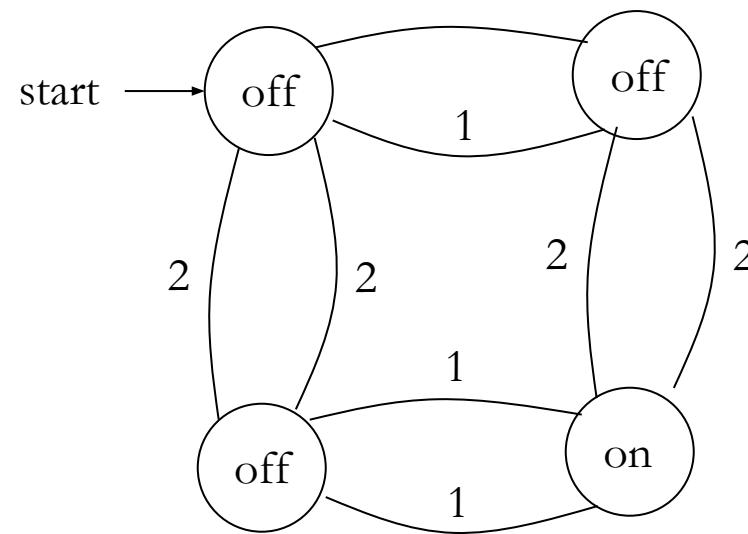
Example:



actions: 1 for “flip switch 1”

actions: 2 for “flip switch 2”

states: on, off



bulb is on if and only if both switches were flipped an odd number of times

Types of Automata

finite automata Devices with a finite amount of memory.
 Used to model “small” computers.

push-down
automata Devices with infinite memory that can be
 accessed in a restricted way.
 Used to model parsers, etc.

Turing
Machines Devices with infinite memory.
 Used to model any computer.

Alphabets and strings

A common way to talk about words, number, pairs of words, etc. is by representing them as strings

To define strings, we start with an alphabet

An **alphabet** is a finite set of symbols.

Examples:

$\Sigma_1 = \{a, b, c, d, \dots, z\}$: the set of letters in English

$\Sigma_2 = \{0, 1, \dots, 9\}$: the set of (base 10) digits

$\Sigma_3 = \{a, b, \dots, z, \#\}$: the set of letters plus the special symbol #

$\Sigma_4 = \{ (,) \}$: the set of open and closed brackets

Strings

A **string** over alphabet Σ is a finite sequence of symbols in Σ .

The empty string will be denoted by e

Examples:

abfbz is a string over $\Sigma_1 = \{a, b, c, d, \dots, z\}$

9021 is a string over $\Sigma_2 = \{0, 1, \dots, 9\}$

ab#bc is a string over $\Sigma_3 = \{a, b, \dots, z, \#\}$

))()() is a string over $\Sigma_4 = \{ (,) \}$

Languages

A **language** is a set of strings over an alphabet.

Languages can be used to describe problems with “yes/no” answers, for example:

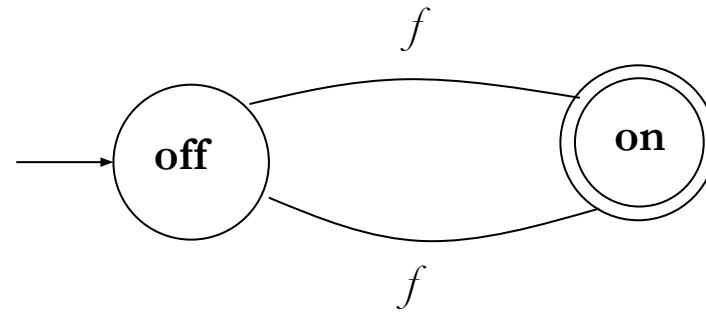
L_1 = The set of all strings over Σ_1 that contain the substring “SRM”

L_2 = The set of all strings over Σ_2 that are divisible by 7 = {7, 14, 21, ...}

L_3 = The set of all strings of the form s#s where s is any string over {a, b, ..., z}

L_4 = The set of all strings over Σ_4 where every (can be matched with a subsequent)

Finite Automata



There are states off and on, the automaton starts in off and tries to reach the “good state” on
What sequences of fs lead to the good state?

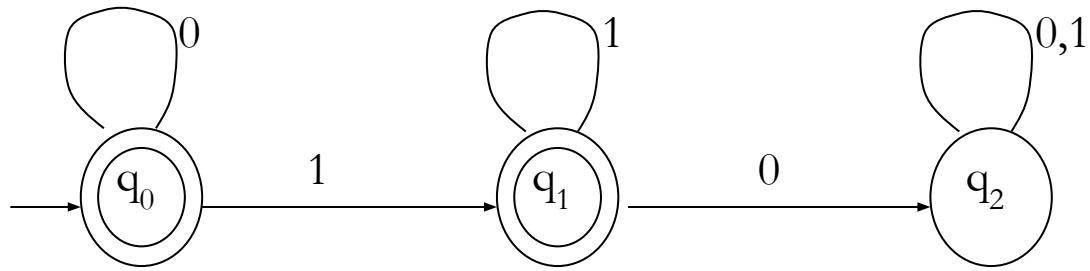
Answer: $\{f, fff, fffff, \dots\} = \{f^n : n \text{ is odd}\}$

This is an example of a deterministic finite automaton over alphabet $\{f\}$

Deterministic finite automata

- A deterministic finite automaton (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where
 - Q is a finite set of states
 - Σ is an alphabet
 - $\delta: Q \times \Sigma \rightarrow Q$ is a transition function
 - $q_0 \in Q$ is the initial state
 - $F \subseteq Q$ is a set of accepting states (or final states).
- In diagrams, the accepting states will be denoted by double loops

Example



alphabet $\Sigma = \{0, 1\}$

start state $Q = \{q_0, q_1, q_2\}$

initial state q_0

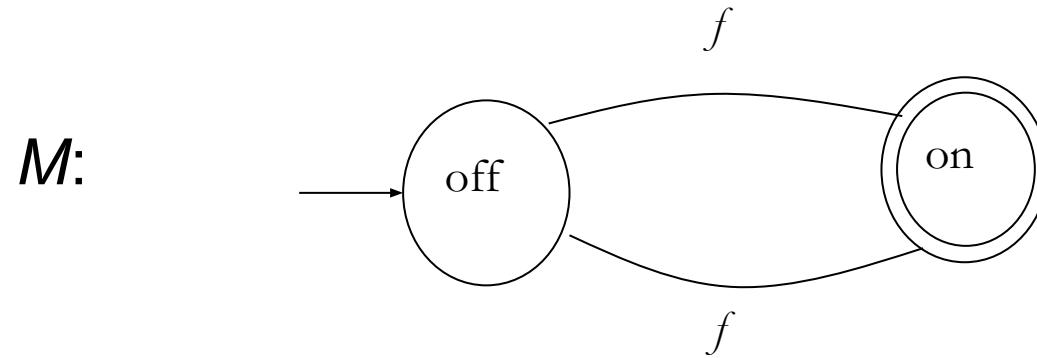
accepting states $F = \{q_0, q_1\}$

transition function δ :

inputs		
states	0	1
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_2	q_2

Language of a DFA

The **language of a DFA** $(Q, \Sigma, \delta, q_0, F)$ is the set of all strings over Σ that, starting from q_0 and following the transitions as the string is read left to right, will reach some accepting state.

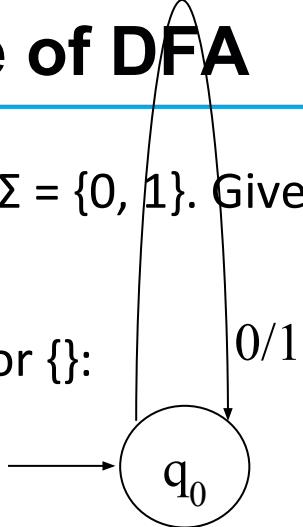


- Language of M is $\{f, fff, fffff, \dots\} = \{f^n : n \text{ is odd}\}$

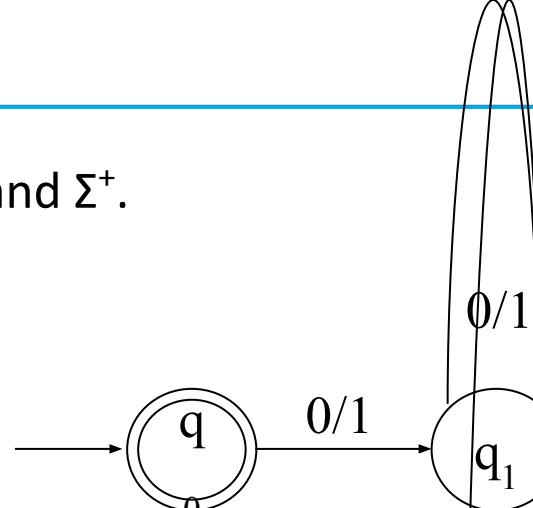
Example of DFA

1. Let $\Sigma = \{0, 1\}$. Give DFAs for $\{\}$, $\{\epsilon\}$, Σ^* , and Σ^+ .

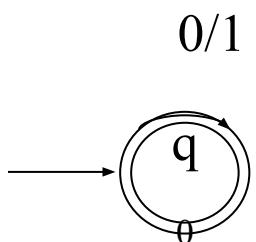
For $\{\}$:



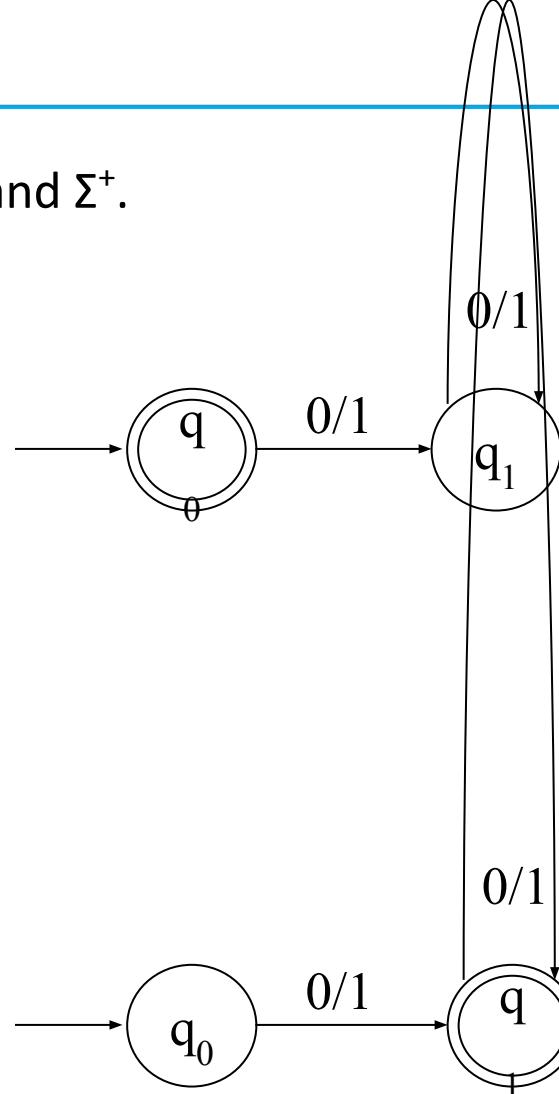
For $\{\epsilon\}$:



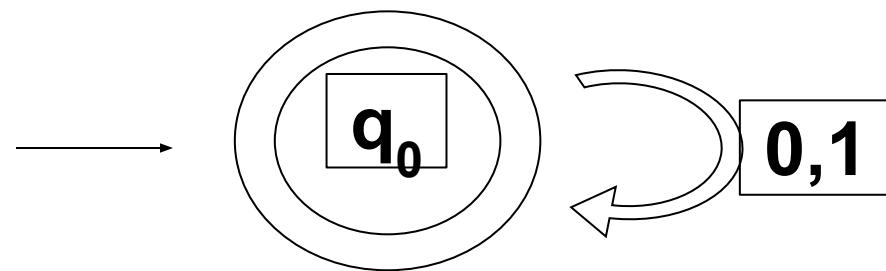
For Σ^* :



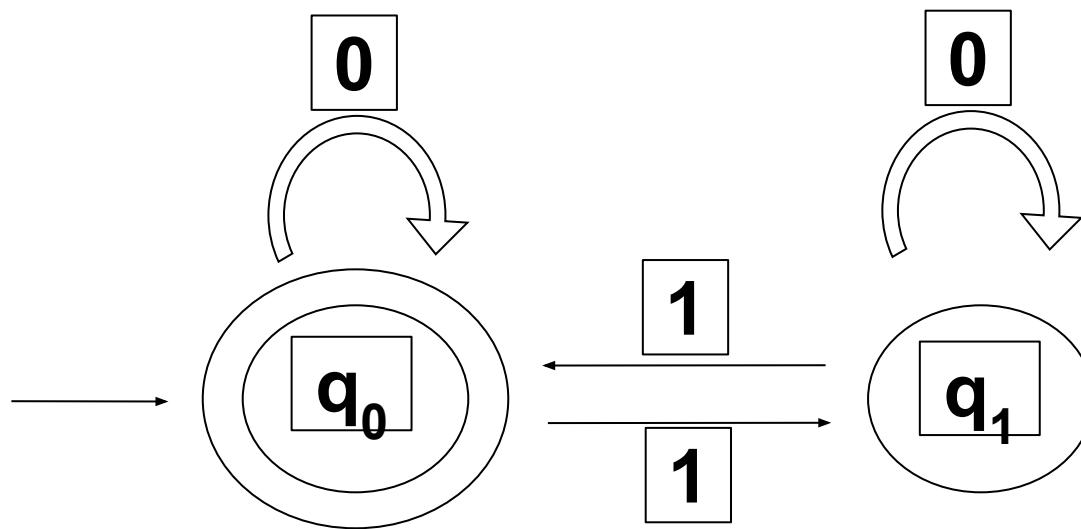
For Σ^+ :



Example of DFA



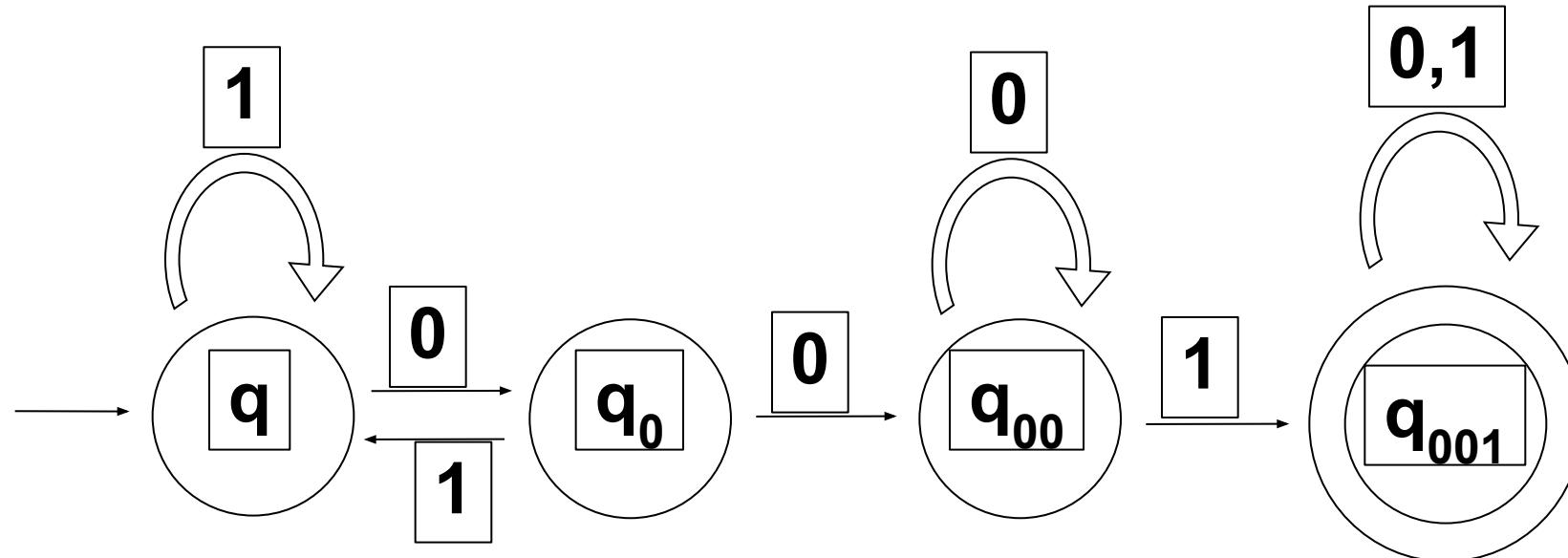
$$L(M) = \boxed{\{0,1\}^*}$$



$$L(M) = \boxed{\{ w \mid w \text{ has an even number of 1s}\}}$$

Example of DFA

Build an automaton that accepts all and only those strings that contain 001



Example of DFA using Python

```
from automata.fa.dfa import DFA
# DFA which matches all binary strings ending in an odd number of '1's
dfa = DFA(
    states={'q0', 'q1', 'q2'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q0', '1': 'q1'},
        'q1': {'0': 'q0', '1': 'q2'},
        'q2': {'0': 'q2', '1': 'q1'}
    },
    initial_state='q0',
    final_states={'q1'}
)
dfa.read_input('01') # answer is 'q1'
dfa.read_input('011') # answer is error
print(dfa.read_input_stepwise('011'))
Answer # yields:
# 'q0'  # 'q0'  # 'q1'
# 'q2'  # 'q1'
```

```
if dfa.accepts_input('011'):
    print('accepted')
else:
    print('rejected')
```

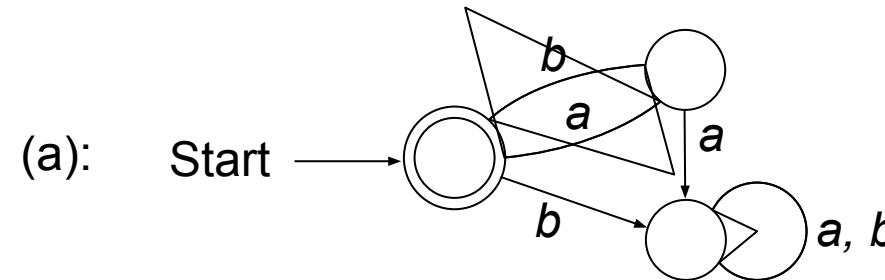
Questions for DFA

Find an DFA for each of the following languages over the alphabet $\{a, b\}$.

- (a) $\{(ab)^n \mid n \in \mathbb{N}\}$, which has regular expression $(ab)^*$.

Solution

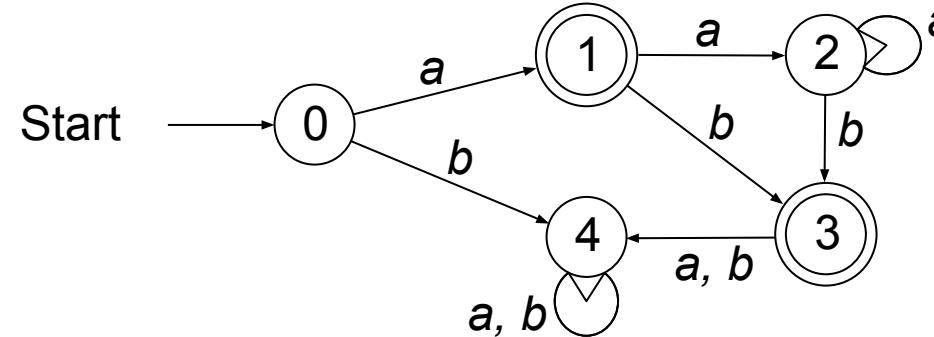
:



- b) Find a DFA for the language of $a + aa^*b$.

Solution

:



Questions for DFA

c) A DFA that accepts all strings that contain 010 or do not contain 0.

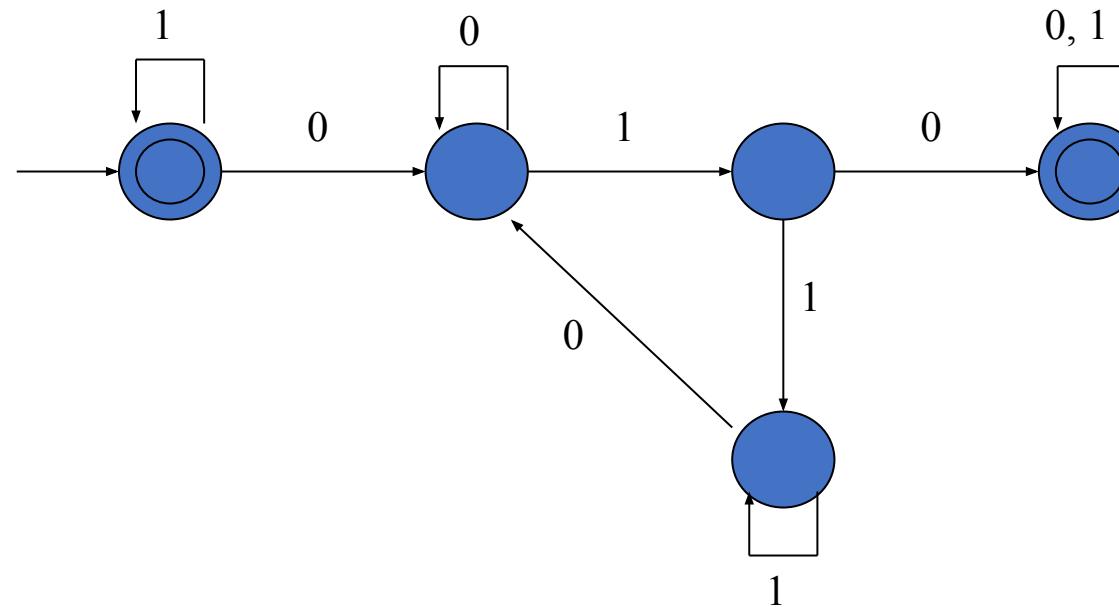
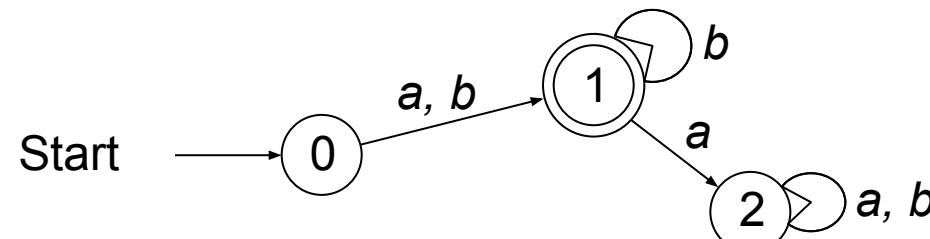


Table Representation of a DFA

A DFA over A can be represented by a transition function $T : \text{States} \times A \rightarrow \text{States}$, where $T(i, a)$ is the state reached from state i along the edge labelled a , and we mark the start and final states. For example, the following figures show a DFA and its transition table.



T	a	b
start	0	1 1
final	1	2 1
2	2	2

Sample Exercises - DFA

1. Write a automata code for the Language that accepts all and only those strings that contain 001
2. Write a automata code for $L(M) = \{ w \mid w \text{ has an even number of } 1s\}$
3. Write a automata code for $L(M) = \{0,1\}^*$
4. Write a automata code for $L(M) = a + aa^*b$.
5. Write a automata code for $L(M) = \{(ab)^n \mid n \in \mathbb{N}\}$
6. Write a automata code for Let $\Sigma = \{0, 1\}$.

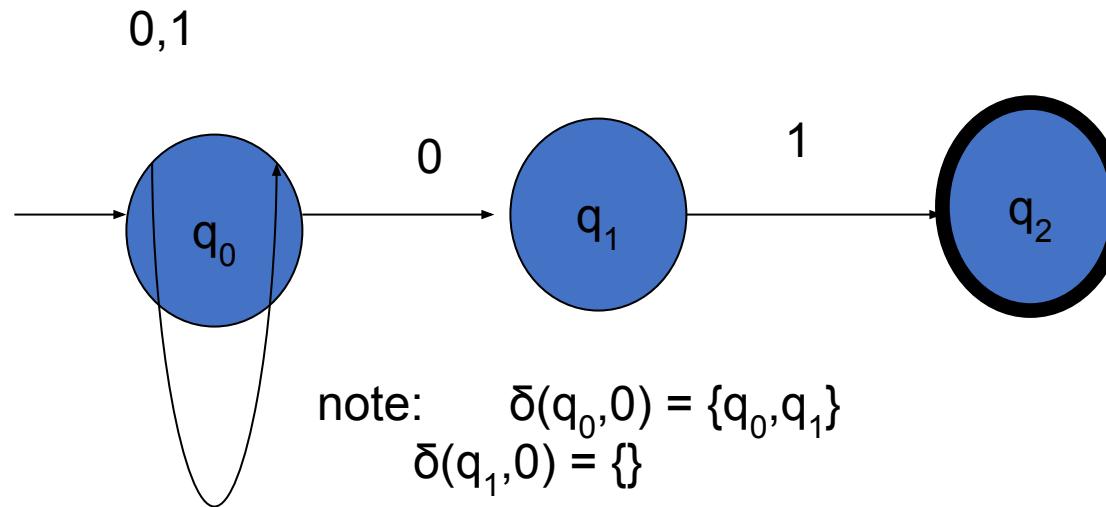
Given DFAs for $\{\}$, $\{\epsilon\}$, Σ^* , and Σ^+ .

NDFA

- A nondeterministic finite automaton M is a five-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where:
 - Q is a finite set of states of M
 - Σ is the finite input alphabet of M
 - $\delta: Q \times \Sigma \rightarrow \text{power set of } Q$, is the state transition function mapping a state-symbol pair to a subset of Q
 - q_0 is the start state of M
 - $F \subseteq Q$ is the set of accepting states or final states of M

Example NDFA

- NFA that recognizes the language of strings that end in 01

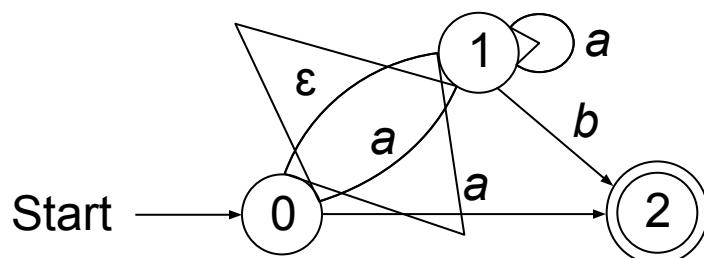


Exercise: Draw the complete transition table for this NFA

NDFA

A nondeterministic finite automaton (NFA) over an alphabet A is similar to a DFA except that epsilon-edges are allowed, there is no requirement to emit edges from a state, and multiple edges with the same letter can be emitted from a state.

Example. The following NFA recognizes the language of $a + aa^*b + a^*b$.



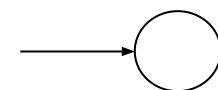
	T	a	b	ϵ
start	0	{1, 2}	\emptyset	{1}
	1	{1}	{2}	\emptyset
final	2	\emptyset	\emptyset	\emptyset

Table representation of NFA

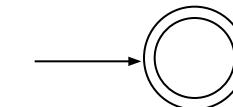
An NFA over A can be represented by a function $T : \text{States} \times A \cup \{\lambda\} \rightarrow \text{power}(\text{States})$, where $T(i, a)$ is the set of states reached from state i along the edge labeled a , and we mark the start and final states. The following figure shows the table for the preceding NFA.

Examples

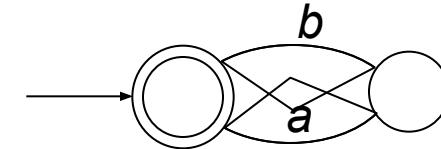
Solutions: (a): Start



(b): Start

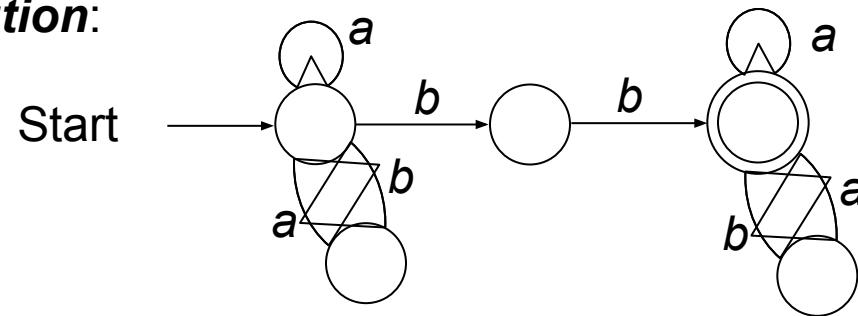


(c): Start



Find an NFA to recognize the language $(a + ba)^*bb(a + ab)^*$.

A solution:



Examples

Algorithm: *Transform a Regular Expression into a Finite Automaton*

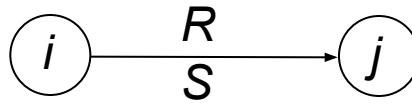
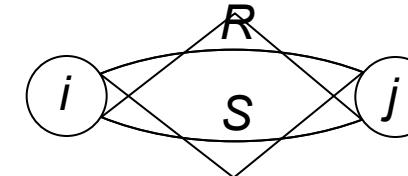
Start by placing the regular expression on the edge between a start and final state:



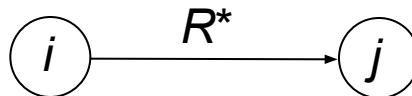
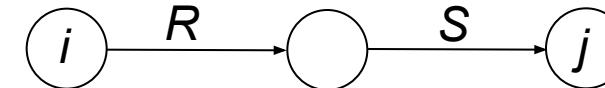
Apply the following rules to obtain a finite automaton after erasing any \emptyset -edges.



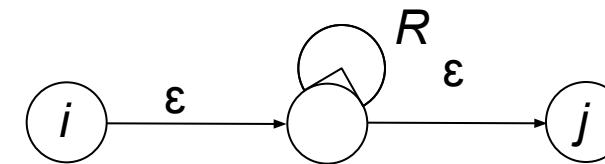
transforms
to



transforms
to

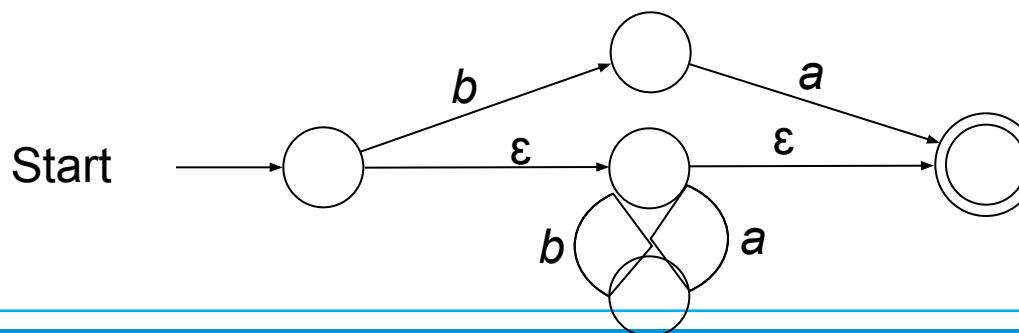


transforms
to



Quiz. Use the algorithm to construct a finite automaton for $(ab)^* + ba$.

Answer:



Example of NFA using Python

```
from automata.fa.nfa import NFA
# NFA which matches strings beginning with 'a', ending with 'a', and
containing
# no consecutive 'b's
nfa = NFA(
    states={'q0', 'q1', 'q2'},
    input_symbols={'a', 'b'},
    transitions={
        'q0': {'a': {'q1'}},
        # Use " as the key name for empty string (lambda/epsilon)
transitions
        'q1': {'a': {'q1'}, '': {'q2'}},
        'q2': {'b': {'q0'}}
    },
    initial_state='q0',
    final_states={'q1'}
)
```

```
nfa.read_input('aba')
ANSWER :{'q1', 'q2'}
```



```
nfa.read_input('abba')
ANSWER: ERROR
```



```
nfa.read_input_stepwise('aba')
if nfa.accepts_input('aba'):
    print('accepted')
else:
    print('rejected')
ANSWER: ACCEPTED
nfa.validate()
ANSWER: TRUE
```

Sample Exercises - NFA

1. Write a automata code for the Language that accepts all end with 01
2. Write a automata code for $L(M) = a + aa^*b + a^*b$.
3. Write a automata code for Let $\Sigma = \{0, 1\}$.

Given NFAs for $\{\}$, $\{\epsilon\}$, $\{(ab)^n \mid n \in \mathbb{N}\}$, which has regular expression $(ab)^*$.

Dependent Types Paradigms

Dependent Types Paradigms

Unit-IV (15 Session)

Session 6-10 cover the following topics:-

- *Dependent Type Programming Paradigm*- S6-SLO1
- *Logic Quantifier: for all, there exists*- S6-SLO2
- *Dependent functions, dependent pairs*– S7-SLO 1
- *Relation between data and its computation*– S7-SLO 2
- *Other Languages: Idris, Agda, Coq* S8-SLO 1
- Demo: Dependent Type Programming in Python S8-SLO2

Lab 11: Dependent Programming (Case Study) (S8)

Assignment : Comparative study of Dependent programming in Idris, Agda, Coq

TextBook:

- 1) Amit Saha, Doing Math with Python: Use Programming to Explore Algebra, Statistics,Calculus and More, Kindle Edition, 2015

URL :

- <https://tech.peoplefund.co.kr/2018/11/28/programming-paradigm-and-python-eng.html>
- <https://freecontent.manning.com/a-first-example-of-dependent-data-types/>

Introduction

- A constant problem:
- Writing a correct computer program is hard
- Proving that a program is correct is even harder
- Dependent Types allow us to write programs and know they are correct before running them.

What is correctness?

- What does it mean to be “correct”?
- Depends on the application domain, but could mean one or more of:
 - **Functionally correct** (e.g. arithmetic operations on a CPU)
 - **Resource safe** (e.g. runs within memory bounds, no memory leaks, no accessing unallocated memory, no deadlock. . .)
 - **Secure** (e.g. not allowing access to another user’s data)

What is Type?

- In **programming**, types are a means of classifying values
- Exp: values 94, "thing", and [1,2,3,4,5] classified as an integer, a string, and a list of integers
- For a ***machine***, types describe how bit patterns in memory are to be interpreted.
- For a ***compiler or interpreter***, types help ensure that bit patterns are interpreted consistently when a program runs.
- For a ***programmer***, types help name and organize concepts, aiding documentation and supporting interactive editing environments.

Introductions

- In computer science and logic, a dependent type is a type whose definition depends on a value.
- It is an overlapping feature of type theory and type systems.
- Used to encode logic's quantifiers like "for all" and "there exists".
- Dependent types may help reduce bugs by enabling the programmer to assign types that further restrain the set of possible implementations.
- Exp: Agda, ATS, Coq, F*, Epigram, and Idris

Dependent Type Example

- **Exp matrix arithmetic**
- **Matrix type** - □ refined it to include the number of rows and columns.
- Matrix 3 4 is the type of 3×4 matrices.
- In this type, 3 and 4 are ordinary values.
- A *dependent type*, such as Matrix, is a type that's calculated from some other values.
- In other words, it *depends on* other values.
- **Definition**
 - A data type is a type which is computed from a *dependent* other value.

Elements of dependent types

- **Dependent functions**
 - The return type of a dependent function may depend on the *value* (not just type) of one of its arguments
 - For instance, a function that takes a positive integer n may return an array of length n , where the array length is part of the type of the array.
 - (Note that this is different from [polymorphism](#) and [generic programming](#), both of which include the type as an argument.)
- **Dependent pairs**
 - A dependent pair may have a second value of which the type depends on the first value

Formal definition

Π type [edit]

Loosely speaking, dependent types are similar to the type of an indexed family of sets. More formally, given a type $A : \mathcal{U}$ in a universe of types \mathcal{U} , one may have a **family of types** $B : A \rightarrow \mathcal{U}$, which assigns to each term $a : A$ a type $B(a) : \mathcal{U}$. We say that the type $B(a)$ varies with a .

A function whose type of return value varies with its argument (i.e. there is no fixed codomain) is a **dependent function** and the type of this function is called **dependent product type**, **pi-type** or **dependent function type**.^[3] For this example, the dependent function type is typically written as

$$\prod_{x:A} B(x)$$

or

$$\prod_{x:A} B(x).$$

If $B : A \rightarrow \mathcal{U}$ is a constant function, the corresponding dependent product type is equivalent to an ordinary **function type**. That is, $\prod_{x:A} B$ is judgmentally equal to $A \rightarrow B$ when B does not depend on x .

Formal definition

Σ type [edit]

The dual of the dependent product type is the **dependent pair type**, **dependent sum type**, **sigma-type**, or (confusingly) **dependent product type**.^[3] Sigma-types can also be understood as **existential quantifiers**. Continuing the above example, if, in the universe of types \mathcal{U} , there is a type $A : \mathcal{U}$ and a family of types $B : A \rightarrow \mathcal{U}$, then there is a dependent pair type

$$\sum_{x:A} B(x).$$

The dependent pair type captures the idea of an ordered pair where the type of the second term is dependent on the value of the first. If

$$(a, b) : \sum_{x:A} B(x),$$

then $a : A$ and $b : B(a)$. If B is a constant function, then the dependent pair type becomes (is judgementally equal to) the **product type**, that is, an ordinary Cartesian product $A \times B$.

Pseudo-code

- **General Code**

```
float myDivide(float a, float b)
{ if (b == 0)
return ???;
Else
return a / b;
}
```

- **Dependent Type Code**

```
float myDivide3
(float a, float b, proof(b != 0)
p)
{
return a / b;
}
```

Auto Checking done here

Python Simple Example

```
from typing import Union def return_int_or_str(flag: bool) ->
    Union[str, int]:
    if flag:
        return 'I am a string!'
    return 0
```

Dependent Type

- » pip install mypy typing_extensions
- from typing import overload
- from typing_extensions import Literal

Literal

Literal type represents a specific value of the specific type.

```
from typing_extensions import Literal
def function(x: Literal[1]) -> Literal[1]:
    return x
function(1)
# => OK!
function(2)
# => Argument has incompatible type "Literal[2]"; expected "Literal[1]"
```

Python Example

x :: Iterator[T1]

y :: Iterator[T2]

z :: Iterator[T3]

product(x, y) :: Iterator[Tuple[T1, T2]]

product(x, y, z) :: Iterator[Tuple[T1, T2, T3]]

product(x, x, x, x) :: Iterator[Tuple[T1, T1, T1, T1]]

- All the above replaced by
 - def product(*args :Tuple[n]) -> Iterator[Tuple[n]]: pass

A first example: classifying vehicles by power source IDRIS Example

Listing 1 Defining a dependent type for vehicles, with their power source in the type (vehicle.idr)

```
data PowerSource = Petrol | Pedal          ①  
data Vehicle : PowerSource -> Type where  
  Bicycle : Vehicle Pedal                ②  
  Car : (fuel : Nat) -> Vehicle Petrol  ③  
  Bus : (fuel : Nat) -> Vehicle Petrol   ④
```

- ① An enumeration type describing possible power sources for a vehicle
- ② A Vehicle's type is annotated with its power source
- ③ A vehicle powered by pedal
- ④ A vehicle powered by petrol, with a field for current fuel stocks

IDRIS Second Example

Listing 2 Reading and updating properties of Vehicle

wheels : Vehicle power -> Nat ①

wheels Bicycle = 2 wheels (Car fuel) = 4 wheels (Bus fuel) = 4

refuel : Vehicle Petrol -> Vehicle Petrol ②

refuel (Car fuel) = Car 100 refuel (Bus fuel) = Bus 200

① Use a type variable, power, because this function works for all possible vehicle types.

② Refueling only makes sense for vehicles that carry fuel.
Restrict the input and output type to Vehicle Petrol.

References

- <http://www.cs.ru.nl/dtp11/slides/brady.pdf>
- <https://freecontent.manning.com/a-first-example-of-dependent-data-types/>
- https://en.wikipedia.org/wiki/Dependent_type
- <https://livebook.manning.com/book/type-driven-development-with-idris/chapter-1/13>
- <https://github.com/python/mypy/issues/366>

UNIT V

PART A: MULTIPLE CHOICE QUESTIONS

Symbolic:

1. Which of the following is false about sympy? (CLO5-L1)
 - a. Sympy is a python library for symbolic mathematics
 - b. It requires external libraries for execution
 - c. It is an alternative to the systems like mathematica or maple

Ans: B

2. Symbols can now be manipulated using some of python operators using _____.(CLO5-L1)
A)+. B) && C) ? D\$

Ans: A

3. Which of the following belongs to the numerical type of sympy (CLO5-L1)
 - a. Float
 - b. Integer
 - c. Decimal
 - d. Factorial

Ans: B

4. Choose the correct output for the following code?(CLO5-L2)

```
Import sympy as sym
```

```
a= sym.Rational(4,6)
```

```
print a
```

a. 6/4

b. 0.66

c. 4/6

d. 1.5

Ans: C

5. What is the output for the trigonometry function? (CLO5-L2)

```
Sym.expand(sym.cos(x+y),trig=true)
```

a. sin(x)*sin(y)+cos(x)*cos(y)

b. sin(x)*cos(y)+cos(x)*sin(y)

c. -sin(x)*sin(y) - cos(x)*cos(y)

d. -sin(x)*sin(y) + cos(x)*cos(y)

Ans: D

6. Differentiate the Sympy Expression using the syntax (CLO5-L1)

a. diff (var,func)

b. diff(func,var)

c. diff(expr,var)

d. diff(var,point)

Ans: B

7. SymPy is able to solve algebraic equations, in one and several variables using(CLO5-L1)

- a. solveset()
- b. series()
- c. real()
- d. limit()

Ans: A

8. Which of the following belongs to the calculus function? (CLO5-L1)

- i. Limit
 - ii. Series
 - iii. Arithmetic
 - iv. Computation
- a. Both i,ii
 - b. Both ii,iii
 - c. Both ii,iv
 - d. Both i,iii,iv

Ans: A

9. Choose the output for the following code? (CLO5-L2)

Limit (sin(x), x,0)

- a. 0
- b. 1
- c. Infinite
- d. Error

Ans: B

10. Which of the following is the correct output for the below given code? (CLO5-L2)

Sym.integrate(x**3, (x,-1,1)

- a. 1
- b. 3
- c. 0
- d. -1

Ans: C

11. Which of the following is the correct output for the below given code? (CLO5-L2)

x,y=sym.symbols('x,y')

A=sym.Matrix([[1,x],[y,1]])

Print A

- a. Matrix ([[1,x], [y,1]])
- b. Matrix ([[x,1],[1,y]])
- c. Matrix ([[0,x], [y,0]])
- d. Matrix ([[x,0],[0,y]])

Ans: A

12. Choose the output for the following code? (CLO5-L2)

Sym.solveset(x**4-1,x)

- a. {1,-1,I,-I}
- b. {-1,1,-I,I}
- c. {0,1,I,-I}
- d. {1,-1,-I,I}

Ans: B

13. Limit the Sympy Expression using the syntax (CLO5-L1)
- a. limit (var,func,point)
 - b. limit(func,var,point)
 - c. limit(func,var)
 - d. limit(var,point)

Ans: B

14. Finite state machines are used for____(CLO5-L1)

- a. Pseudo random test patterns
- b. Deterministic test patterns
- c. Random test patterns
- d. Algorithmic test patterns

Ans:D

15. According to the given transitions, which among the following are the epsilon closures of q1 for the given NFA? (CLO5-L3)

$$\Delta(q_1, \varepsilon) = \{q_2, q_3, q_4\}$$

$$\Delta(q_4, 1) = q_1$$

$$\Delta(q_1, \varepsilon) = q_1$$

- a. q4
- b. q2
- c. q1
- d. q1, q2, q3, q4

Ans: D

16. Which of the following tuples order are correct for Deterministic Finite Automata (DFA) (CLO5-L1)

- a. (Q, S, d, q_0, F)
- b. (Q, S, d, F, q_0)
- c. (S, Q, d, q_0, F)
- d. (Q, S, q_0, d, F)

Ans: A

17. NFA, is called as 'non-deterministic' because of :(CLO5-L1)

- a. The result is undetermined
- b. The choice of path is non-deterministic
- c. The state to be transited next is non-deterministic
- d. All of the mentioned

Ans: B

18. _____ is a class attribute defined by its source state and destination state. .(CLO5-L1)

- a. LGPL
- b. Scipy
- c. Transition
- d. State

Ans : C

19. Among the following which is used to define the behavior of what we want to achieve (CLO5-L1)

- a. State
- b. Automata
- c. Transition
- d. Machine

Ans: D

20. _____is a triplet consisting of two states and one command needed for the change of one state to the other. .(CLO5-L1)

- a. machine
- b. Automata
- c. State
- d. Transitions

Ans:D

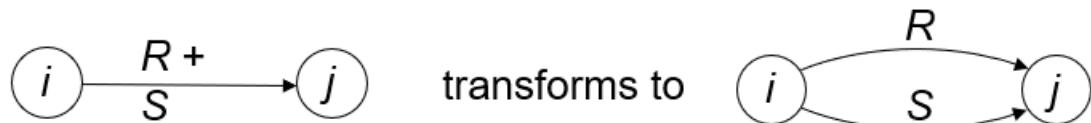
21. _____is a mathematical model of computation. .(CLO5-L1)

- a. state machine
- b. Automata
- c. State
- d. Transitions

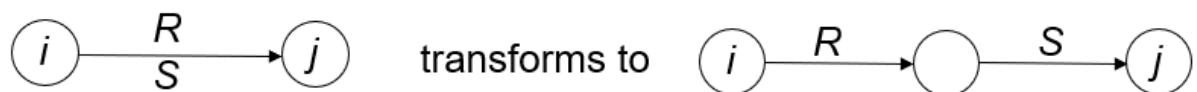
Ans: A

22. Which of the following is correct among the following expressions? (CLO5-L2)

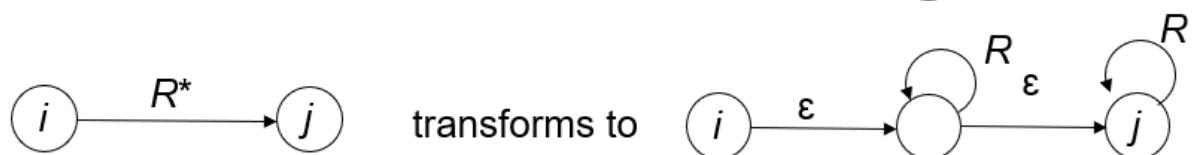
i.



ii.



iii.



- a. Both i,ii
- b. Both ii,iii
- c. Both i,iii
- d. All of the above

Ans: A

23. What kind of abstract machine can recognize strings in a regular set? (CLO5-L1)

- a. DFA
- b. NFA
- c. PDA
- d. None of the above

Ans: A

24. Which of the following statements is wrong? (CLO5-L1)

- a. The language accepted by finite automata are the languages denoted by regular expressions
- b. For every DFA there is a regular expression denoting its language
- c. For a regular expression r , there does not exist NFA with $L(r)$ any transit that accept
- d. None of the above.

Ans: C

25. Let for $\Sigma = \{0,1\}$ $R = (\Sigma\Sigma\Sigma)^*$, the language of R would be-----
(CLO5-L2)

- a. $\{w \mid w \text{ is a string of odd length}$
- b. $\{w \mid w \text{ is a string of length multiple of 3}\}$
- c. $\{w \mid w \text{ is a string of length 3}\}$
- d. All of the above.

Ans: B

26. $(a+b)^*$ is equivalent to----- (CLO5-L2)

- a. b^*a^*
- b. $(a^*b^*)^*$
- c. a^*b^*
- d. None of the above.

Ans: B

27. In regular expressions, the operator '*' stands for----- (CLO5-L1)

- a. Concatenation
- b. Addition
- c. Selection
- d. Iteration

Ans: D

28. The regular expression with all strings of 0's and 1's with at least two consecutive 0's is----- (CLO5-L2)

- a. $1 + (10)^*$ yh
- b. $(0+1)^*011$
- c. $(0+1)^*00(0+1)^*$
- d. $0^*1^*2^*$

Ans: C

29. Which of the following operation can be applied on regular expressions? (CLO5-L1)

- a. Union
- b. Concatenation
- c. Closure
- d. All of the above

Ans: D

30. Finite state machine will initially set to all zeroes. (CLO5-L1)

- a. True
- b. False

Ans: A

31. _____ allow millions of different machines, using all sorts of different network hardware, to pass packets to each other over the fabric of an IP network.(CLO5-L1)

- a)Sockets

- b)Client machine
- c)server machines
- d)IP address

Ans: d

32. Which are all necessary to direct a packet to its destination.(CLO5-L2)

- a)IP Address
- b)IP address and port
- c)port
- d)InternetAssignedNumbers

Ans:b

33. _____ (0–1023) are for the most important and widely-used protocols(CLO5-L1)

- a. Well Known ports
- b)Registered ports
- c)Known ports
- d)Unregistered ports

Ans:a

34. _____ (1024–49151) are not usually treated as special by operating systems(CLO5-L1)

- a)Registered ports
- b)well known ports
- c)Known ports
- d)Unregistered ports

Ans:a

35. Python's standard socket module supports _____(CLO5-L1)

- a)getByName()
- b)getName()
- c)gethostbyname()
- d)getServName()

Ans:c

36. Example for Connection oriented protocol(CLO5-L1)

- a)UDP
- b)SMTP
- c)FTP
- d)TCP

Ans:d

37. Example for Connection less protocol(CLO5-L1)

- a)SMTP
- b)FTP
- c)UDP
- d)TCP

Ans:c

38. _____ is an application-level block of transmitted data.(CLO5-L1)

- a)data
- b)datagram
- c)segmentation
- d)Fragmentation

Ans:b

39. Simple server uses _____ command to request a UDP network address.(CLO5-L1)

- a)bind()
- b)socket()
- c)getName()
- d)getsockName()

Ans:a

40. Which method is used by python program to retrieve the current IP and port to which the socket is bound. (CLO5-L2)

- a)bind()
- b)getsockname()
- c)getName()
- d)getSocket()

Ans:b

41. Sending packets with another computer's return address is called _____(CLO5-L1)

- a)Binding
- b) Unification
- c)Spoofing
- d) IP address

Ans:c

42. The concept by which larger UDP packets are spitted into several small physical packets(CLO5-L1)

- a)fragmentation
- b)Binding
- c) Unification
- d) partition

Ans:a

43. The MTU is _____ that all of the network devices between two hosts will support. (CLO5-L1)

- a)smallest packet size
- b)medium packet size
- c)average packet size
- d)largest packet size

Ans:d

44. _____ is efficient only if your host ever only sends one message at a time, then waits for a response.(CLO5-L1)

- a)UDP
- b)TCP
- c)FTP
- d)SMTP

Ans: a

45. Which protocol provides reliable connection?(CLO5-L2)

- a)UDP
- b) FTP
- c)TCP
- d)SMTP

Ans:c

46. TCP uses a _____ that counts the number of bytes transmitted.(CLO5-L1)

- a) protocol
- b)counter
- c)rules
- d)Ports

Ans:b

47. The amount of data that a sender is willing to have on the wire at any given moment is called _____(CLO5-L1)
a) The size of the TCP window. b) counter c) port d) protocol

Ans:a

48. TCP uses _____ to distinguish different applications running at the same IP address, and follows exactly the same conventions regarding well-known and ephemeral port numbers.(CLO5-L1)
a) IP address b)port c)port numbers d) socket ID
Ans:c

49. _____refers to the address family ipv4.(CLO5-L1 (CLO5-L1))
a) Af_INET b) AS_INET c) AG_INET d) AN_INET

Ans:a

50. A server has a _____method which puts the server into listen mode.(CLO5-L1)
a)list() b)listento() c)listen() d)listenfrom()

Ans:c

51. _____ are the end-point of a two-way communication .(CLO5 L1) a)IP Address b)sockets c)sockets ID d) Object ID
Ans:b

52. Identify the method used to connect the client to host and port and initiate the connection towards the server.(CLO5-L3)
a)sock_object.recv() b)sock_object.send()
c)sock_object.append() d)sock_object.connect():
Ans:d

53. Identify the method to receive messages at endpoints when the value of the protocol parameter is TCP.(CLO5-L3)
a)sock_object.recv() b)sock_object.send()
c)sock_object.append() d)sock_object.connect():
Ans:a

54. Which method is used to receive messages at endpoints if the protocol used is UDP.(CLO5-L2)
a)sock_object.recv() b)sock_object.recvfrom()
c)sock_object.append() d)sock_object.connect():
Ans:b

55. Identify the method that returns host name.(CLO5-L1)

- a)sock_object.recv() b)sock_object.recvfrom()
c)sock_object.gethostname d)sock_object.connect():
Ans:c

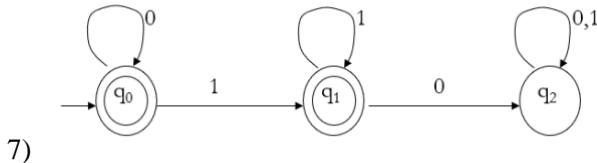
56. Identify the method to send messages from endpoints if the protocol parameter is UDP.(CLO5-L3)
a)sock_object.sendto() b)sock_object.recvfrom()
c)sock_object.gethostname d)sock_object.connect():
Ans:a

57. _____can be used to end direction of communication in a socket(CLO5-L1)
a. The shutdown() call b)Shut() c)close() d)end()

Ans:c

PART: B (4MARKS)

- 1) Write a program to factorize the following expression. (CLO5-L3)
 $x^{**3} + 3*x^{**2}*y + 3*x*y^{**2} + y^{**3}$
- 2) What is SymPy? (CLO5-L1)
- 3) Write the commands to perform the operations on substitutions and expressions (CLO5-L1)
- 4) Design a DFA that accepts all strings that contain 010 or do not contain 0. (CLO5-L2)
- 5) Differentiate Finite state machine and Non deterministic Finite state machine. (CLO5-L2)
- 6) Let $\Sigma = \{0, 1\}$. Give DFAs for $\{\}$, $\{\epsilon\}$, Σ^* , and Σ^+ . (CLO5-L2)



- 7) Provide the DFA function for the above-mentioned diagram (CLO5-L2)

- 8) Compare the features of LISP and Wolfram. (CLO5-L2)
- 9) Write a DFA automata code for $L(M) = \{ w \mid w \text{ has an even number of } 1s \}$ (CLO5-L2)
- 10) Write a DFA automata code for $L(M) = \{(ab)^n \mid n \in \mathbb{N}\}$ (CLO5-L2)
- 11) Construct NFA that recognizes the language of strings that end in 01 (CLO5-L2)
- 12) Find an NFA to recognize the language $(a + ba)^*bb(a + ab)^*$. (CLO5-L2)
- 13) Write the features of GUI programming. (CLO5-L1)
- 14) Explain briefly Automata based programming in python. (CLO5-L1)
- 15) Write the syntax for Expand and Factor command. Give example. (CLO5-L1)
- 16) Give example for DFA and explain. (CLO5-L2)
- 17) Differentiate Python and Jython. (CLO5-L2)
- 18) Write the syntax for series and Integration command. Give example .(CLO5-L1)
- 19) Give example for NFA and explain. (CLO5-L1)
- 20) Differentiate TCP and UDP(CLO5-L2)
- 21) Compare connection oriented and connectionless service(CLO5-L2)
- 22) Differentiate automatic and manual configuration(CLO5-L2)
- 23) How will you generate random port numbers?(CLO5-L2)
- 24) What is file descriptors? Give example(CLO5-L1)
- 25) Differentiate bind() and getSockName() .(CLO5-L2)
- 26) What is UDP fragmentation? Give example .(CLO5-L1)
- 27) What will happen when you send large packets? Give example.(CLO5-L2)
- 28) Differentiate multicast and broadcast.(CLO5-L2)
- 29) Write the code to connect server and client in TCP.(CLO5-L3)
- 30) How does TCP provides reliable connection.(CLO5-L2)

- 31) Compare passive and active socket.(CLO5-L2)
 32) Write the code to connect server and client in UDP.(CLO5-L3)

PART-C (12 MARKS)

1. Write a program to expand and factorize the following expression. (CLO5-L3)

- a. $x^3 + 3x^2y + 3xy^2 + y^3 = (x + y)^3$
- b. $x + y + x^*y$

2. Consider the following series:

$$X + (X^2/2) + (X^3/3) + (X^4/4) + \dots + (X^n/n)$$

Write a python program that will ask a user to input a number, n, and print this series for that number. In the series, x is a symbol and n are an integer input by the program's user. The nth term in this series is given as (X^n/n) . (CLO5-L3)

- 3. Write a program to implement client server communication using UDP. (CLO5-L3)
 - 4. Write a program to demonstrate the concept of UDP fragmentation and explain .(CLO5-L3)
 - 5. Explain automata-based programming paradigm. (CLO5-L1)
 - 6. Design go, slowdown and stop events according to the traffic scenario to cross the road using python code and also draw transition diagram and transition table for the same scenario. (CLO5-L3)
 - 7. Write a example program to find limits, differentiation, Series and Integration. (CLO5-L1)
 - 8. Differentiate DFA and NFA. Explain NFA with example. (CLO5-L2)
 - 9. Build an automaton that accepts all and only those strings that contain 001 (CLO5-L2)
 - 10. Find an DFA for each of the following languages over the alphabet {a, b}.(CLO5-L3)
 - (a) $\{(ab)^n \mid n \in \mathbb{N}\}$, which has regular expression $(ab)^*$.
 - b) Find a DFA for the language of $a + aa^*b$.
 - 11.a. Write NFA automata code for the Language that accepts all end with 01 (CLO5-L3)
 - b. Write a automata code for $L(M) = a + aa^*b + a^*b$.
 - c Write a automata code for Let $\Sigma = \{0, 1\}$.
- Given NFAs for $\{\}$, $\{\epsilon\}$, $\{(ab)^n \mid n \in \mathbb{N}\}$, which has regular expression $(ab)^*$.
12. Write a program to convert NFA to DFA (CLO5-L2)
13. Explain in detail about communication using UDP with example. .(CLO5-L1)

UNIT 4

Dependent Programming paradigm, parallel and concurrent Programming paradigm

1. Parallelism representation is critical to the success of ----- (CLO-4,L1)

- a)High-performance computing.
- b)Low-performance computing
- c)Scaling
- d)Vectorization

Ans:a

2. Parallel programming through a combination of -----and ----- (CLO-4,L1)

- a.Patterns, examples
- b.Algorithms , flowcharts
- c.Models , methods
- d.Classes ,objects

Ans:a

3.What is multithreaded programming? (CLO-4,L1)

- a) It's a process in which two different processes run simultaneously
- b) It's a process in which two or more parts of same process run simultaneously
- c) It's a process in which many different process are able to access same information
- d) It's a process in which a single process can access information from many sources

Ans:b

4. Which of these are types of multitasking? (CLO-4,L2)

- a) Process based
- b) Thread based
- c) Process and Thread based
- d) None of the mentioned

Ans:c

5.What will happen if two thread of the same priority are called to be processed simultaneously? (CLO-4,L2)

- a) Anyone will be executed first lexicographically
- b) Both of them will be executed simultaneously
- c) None of them will be executed
- d) It is dependent on the operating system

Ans:d

6. Which of these statements is incorrect? (CLO-4,L2)

- a) By multithreading CPU idle time is minimized, and we can take maximum use of it
- b) By multitasking CPU idle time is minimized, and we can take maximum use of it
- c) Two threads in Java can have the same priority
- d) A thread can exist only in two states, running and blocked

Ans:d

7. Identify the technique that allows more than one program to be ready for execution and provides the ability to switch from one process to another. (CLO-4,L2)

- a) multitasking
- b) multiprocessing
- c) multitasking
- d) multiprogramming

Ans:d

8. The technique that increases the system's productivity. (CLO-4,L1)

- a) multiprogramming
- b) multitasking
- c) multiprocessing

d) single-programming

Ans:a

9. _____ is a property which more than one operation can be run simultaneously but it doesn't mean it will be. (CLO-4,L1)

- a. Concurrency
- b. Semaphore
- c. Mutual exclusion
- d. parallel process

Ans:a

10. _____ is a light-weight cooperatively-scheduled execution unit.(CLO-4,L3)

- a. gevent.Greenlet
- b. gevent.spawn()
- c. gevent.spawn_later()
- d. gevent.spawn_raw()

Ans:a

11. Which keyword is used to define methods in Python? (CLO-4,L2)

- (a) function
- (b) def
- (c) method
- (d) All of these

Ans:B

12. What is the correct translation of the following statement into mathematical logic? "Some real numbers are rational" .(CLO-4,L3)

- (A) $\exists x (\text{real}(x) \vee \text{rational}(x))$
- (B) $\forall x (\text{real}(x) \rightarrow \text{rational}(x))$
- (C) $\exists x (\text{real}(x) \wedge \text{rational}(x))$
- (D) $\exists x (\text{rational}(x) \rightarrow \text{real}(x))$

A

B

C

D

Ans: c

13. Which one of the following options is CORRECT given three positive integers x, y and z, and a predicate? .(CLO-4,L3)

$$P(x) = \neg(x=1) \wedge \forall y (\exists z (x=y*z) \Rightarrow (y=x) \vee (y=1))$$

$P(x)$ being true means that x is a prime number

$P(x)$ being true means that x is a number other than 1

$P(x)$ is always true irrespective of the value of x

$P(x)$ being true means that x has exactly two factors other than 1 and x

Ans: a

14. Suppose the predicate $F(x, y, t)$ is used to represent the statement that person x can fool person y at time t. which one of the statements below expresses best the meaning of the formula $\forall x \exists y \exists t (\neg F(x, y, t))$? .(CLO-4,L3)

- (a) Everyone can fool some person at some time
- (b) No one can fool everyone all the time
- (c) Everyone cannot fool some person all the time
- (d) No one can fool some person at some time

Ans: b

15. Which one of the following is the most appropriate logical formula to represent the statement? "Gold and silver ornaments are precious". (CLO-4,L3)

The following notations are used:

$G(x)$: x is a gold ornament

$S(x)$: x is a silver ornament

$P(x)$: x is precious

- (a) $\forall x(P(x) \rightarrow (G(x) \wedge S(x)))$
- (b) $\forall x((G(x) \wedge S(x)) \rightarrow P(x))$
- (c) $\exists x((G(x) \wedge S(x)) \rightarrow P(x))$
- (d) $\forall x((G(x) \vee S(x)) \rightarrow P(x))$

Ans : d

16. Which one of the first order predicate calculus statements given below correctly express the following English statement? .(CLO-4,L3)

Tigers and lions attack if they are hungry or threatened.

- (A) $\forall x [(\text{tiger}(x) \wedge \text{lion}(x)) \rightarrow \{(\text{hungry}(x) \vee \text{threatened}(x)) \rightarrow \text{attacks}(x)\}]$
- (B) $\forall x [(\text{tiger}(x) \vee \text{lion}(x)) \rightarrow \{(\text{hungry}(x) \vee \text{threatened}(x)) \wedge \text{attacks}(x)\}]$
- (C) $\forall x [(\text{tiger}(x) \vee \text{lion}(x)) \rightarrow \{\text{attacks}(x) \rightarrow (\text{hungry}(x) \vee \text{threatened}(x))\}]$
- (D) $\forall x [(\text{tiger}(x) \vee \text{lion}(x)) \rightarrow \{(\text{hungry}(x) \vee \text{threatened}(x)) \rightarrow \text{attacks}(x)\}]$

Ans: d

17. What is the first order predicate calculus statement equivalent to the following? (CLO-4,L3)

Every teacher is liked by some student

- (A) $\forall(x) [\text{teacher}(x) \rightarrow \exists(y) [\text{student}(y) \rightarrow \text{likes}(y, x)]]$
- (B) $\forall(x) [\text{teacher}(x) \rightarrow \exists(y) [\text{student}(y) \wedge \text{likes}(y, x)]]$
- (C) $\exists(y) \forall(x) [\text{teacher}(x) \rightarrow [\text{student}(y) \wedge \text{likes}(y, x)]]$

(D) $\forall (x) [\text{teacher}(x) \wedge \exists (y) [\text{student}(y) \rightarrow \text{likes}(y, x)]]$

Ans: b

- 18.
- I. $\neg \forall x (P(x))$ II. $\neg \exists x (P(x))$
III. $\neg \exists x (\neg P(x))$ IV. $\exists x (\neg P(x))$

Which of the above two are equivalent? (CLO-4,L3)

- (A) I and III
(B) I and IV
(C) II and III
(D) II and IV

Ans: b

19. _____ is a builtin python module where all possible types are defined. .(CLO-4,L2)

- (a) overload
b)typing
c)function
d)literal

Ans: b

20. _____ type represents a specific value of the specific type. .(CLO-4,L1)

- a) overload
b) typing
c) literal
d) None of the above

Ans: c

21. _____ is required to define multiple function declarations with different input types and results. .(CLO-4,L1)

- a) overload
- b) typing
- c) literal
- d) None of the above

Ans: a

PART B:(4 MARKS)

1. State parallel programming paradigm. (CLO-4, L1)
2. Differentiate parallel programming with functional programming. (CLO-4, L2)
3. Explain about Multithreading. (CLO-4, L1)
4. Explain about Multiprocessing. (CLO-4, L1)
5. Relate Serial processing concepts in Python. (CLO-4, L3)
6. Differentiate Serial Processing and Parallel Processing. (CLO-4, L3)
7. Demonstrate Multiprocessing module in Python. (CLO-4, L3)
8. Describe about Process class. (CLO-4, L2)
9. Design a Pool class in Python.(CLO-4, L3)
10. State Concurrent programming paradigm. (CLO-4, L1)
11. Compare multiprocessing and multitasking. (CLO-4, L2)
- 12.What are dependent functions ?(CLO4-L1)
13. Define typing module in dependent type programming? (CLO-4, L2)
- 14.Define dependent functions? (CLO-4, L2)
- 15.What is predicate logic? (CLO-4, L2)
- 16.Different types of quantifiers. (CLO-4, L1)

17.Explain Universal Quantifier and Existential Quantifier (CLO-4, L1)

18.Write some examples of Universal Qunatifier. (CLO-4, L2)

19 Define overload and literal in dependent type programming (CLO-4, L2)

20. Write the syntax for Existential Quantifier. (CLO-4, L2)

PART :C(12 MARKS)

1. Write a python program to implement producer consumer problem. (CLO-4, L3)

2. Implement the concept “Pool class” by importing a package pool. (CLO-4, L3)

3. Write a python program to implement dinning philosopher problem. (CLO-4, L3)

4. Explain the differences between multithreading and multiprocessing with example? (CLO-4, L1)

5. Write a program to implement thread synchronization (CLO-4, L3)

6. Compare Concurrent programming paradigm and functional programming paradigm with example program. (CLO-4, L2)

7. Explain in detail about dependent type programming .(CLO4-L1)

8. Write a python program to create Type aliases using typing module.. (CLO-4, L3)

9. Write a python program to check every **key:value** pair in a dictionary and check if they match the **name@email** format using typing module.. (CLO-4, L3)

10. Write a python program to create new user defined type Student using typing module.. (CLO-4, L3)

11. Write a python program for function dependent type using overload and literals .. (CLO-4, L3)

1. What does a threading.Lock do?

- a) Pass messages between threads
- b) Allow only one thread at a time to access a resource**
- c) Wait until a thread is finished
- d) SHIFT TO ALL CAPS

2. What does the Thread.join() method do?

- a) Waits for the thread to finish**
- b) Restricts access to a resource
- c) Adds the thread to a pool
- d) Merges two threads into one

3) Race conditions are ...

- a) Testing which thread completes first
- b) Two threads incorrectly accessing a shared resource**
- c) The weather on race day
- d) Something you should add to your code

4) It sets the lock state to locked. If called on a locked object, it blocks until the resource is free.

- a) lock()
- b)release()
- c) acquire()**
- d)join()

5) You have thread T1, T2, and T3. How will you ensure that thread T2 is run after T1 and thread T3 after T2?

- a) Sleep method
- b) Join Method**
- c)Release
- d)lock method

6) What are the libraries in Python that support threads?

- _threading
- b) Thread
- c)None
- d)thread**

7) Mention the correct syntax for creating Thread object for calling increment methods

- a) t1 = threading.Thread()
- b) t1 = threading.Thread(target)
- c) t1 = threading.Thread(target=incr)**
- d) t1 = threading.Thr

8) Which leads to concurrency?

- a) Serialization
- b) Parallelism**
- c) Serial processing
- d) Distribution

9) Which is not a method for parallelism?

- a) Message Passing
- b) Shared Memory
- c) Threads
- d) Sockets**

10) A race condition occurs when multiple processes or threads read and write

- a). Input b). Information c). **Data Items** d). Programs

11) For a single processor system, implementation of semaphores is possible to inhibited through

- a) Deadlock b) **Interrupts** c) Lock Step d) Paging

12) Which method controls the execution of thread in python?

- a) Wait b) **Sleep** c) Acquire d) Lock

13) How does run() method is invoked?

- a) By Thread.create() b) **By Thread.start()**

- c) None d) By Thread.run()

14) What is the difference between *threading.Lock* and *threading.RLock*?

- a) Lock and RLock both primitives are owned by a single thread.

- b) **Lock is owned by none while RLock is owned by many.**

- c) Lock is owned by a thread while RLock is owned by many.

- d) Lock and RLock both primitives are owned by many.

15) What is the difference between a *semaphore* and *bounded semaphore*?

- a) Semaphore holds a counter for the number of release() calls minus the number of acquire() calls, plus an initial value but bounded semaphore doesn't.

- b) A semaphore makes sure its current value doesn't exceed its initial value while bounded semaphore doesn't.

- c) **A bounded semaphore makes sure its current value doesn't exceed its initial value while semaphore doesn't.**

- d) Bounded semaphore holds a counter for the number of release() calls minus the number of acquire() calls, plus an initial value but semaphore doesn't.

16) What is the method that wakes up all thread waiting for the condition?

- a) releaseAll() b) notify() c) **notifyAll()** d) release()

17 How to terminate a blocking thread?

- a). **thread.stop() & thread.wait()** b) thread.stop()

- c) thread.terminate() d) None

18. Which synchronization method is used to guard the resources with limited capacity, e.g. a database server?

- a) Event b) Condition c) Lock d) **Semaphore**

19. How to detect the status of a python thread?

- a) isActive()
- b) isDaemon()
- c) None
- d) **isAlive()**

20. Execution of several activities at the same time.

- a) processing
- b) parallel processing**
- c) serial processing
- d) multitasking

1. SymPy is a collection of mathematical algorithms and convenience functions built on the -----extension of Python

- a) numpy
- b) scikit
- c) sys
- d) functools

2. Exponential function computes the -----

- a) 10^{**x} element-wise
- b) 10^{**x} row-wise
- c) 10^{**x} column-wise
- d) 10^*x element-wise

3. _____ evaluates the expression to a floating-point number.

- a) evalf
- b) fval
- c) float
- d) valf

4. what is the output for the following expression $((x+y)^{**2}).expand()$

- a) $x^2 + 2xy + y^2$
- b) $x^2 + 2*x*y + y^2$
- c) $x^2 + 2*x*y + y^2$
- d) $x^{**2} + 2*x*y + y^{**2}$

5. limit($(5^{**x} + 3^{**x})^{**}(1/x)$, x, oo) ,what is the output

- a) 5
- b) oo
- c) 1
- d) 0

6) Higher derivatives can be calculated using the which method

- a) highder(func,var,n)
- b) diff(func, var, n)
- c) diff(n,var,func)
- d) diff(func, var)

7) what is the output

```
>>> x = Symbol('x')
>>> y = Symbol('y')
>>> A = Matrix([[1,x], [y,1]])
>>> A**2
```

- a) $[1, x]$
 $[y, 1]$
- b) $[xy + 1, 2x]$
 $[2y, xy + 1]$
- c) $[x*y + 1, 2*y]$
- d) $[x*y + 1, 2*x]$

[$2*x$, $x*y + 1$]

[$2*y$, $x*y + 1$]

8) .match() method, along with the ----- class, to perform pattern matching on expressions.

- a) pattern
- b) func
- c) wild**
- d) dictionary

9) which among the following function Return or print, respectively, a pretty representation of expr

- a) pretty(expr)
- b) pretty_print(expr)
- c) pprint(expr)
- d) all of the above**

10) What is the output of

```
from sympy.abc import a, b
```

```
expr = b*a + -4*a + b + a*b + 4*a + (a + b)*3
```

- a) $ba-4a+b+ab+4a+3(a+b)$
- b) $2*a*b + 3*a + 4*b$**
- c) $2ab+3a+4b$
- d) all of the above

11) print(pi.evalf(30))

- a) $3.14/30$
- b) $30/3.14$
- c) 3.14159265358979323846264338328**
- d) 3.14

12) which is the correct way to write equation for $x^2=x$ in sympy

- a) $x^{**2} = x$**
- b) $x*x = x$
- c) $x\%2 = x$
- d) $X^2 = x$

13) how to find a solution for a equation in a given interval

- a) solve(equation,range)
- b) equation(solve,range)
- c) solveset()**
- d) both a and b

14) Allows , the same elements can appear multiple times at different positions

a) set **b)sequence**

c) dictionary d) none

15) which is the snippet to find the eigenvalues of $\begin{bmatrix} 1 & 2 \\ 2 & 2 \end{bmatrix}$

- a) `Matrix([[1, 2], [2, 2]]).eigenvals()`
b) `Matrix([[1, 2], [2, 2]]).eigenvalues()`
c) both a and b d) `eigen([[1, 2], [2, 2]])`

16 What is the purpose of sympify() method?

- a. Convert expression of string type to mathematical expression
- b. Convert mathematical expression to String
- c. Convert mathematical expression to character
- d. Convert tuple to mathematical expression

17 Find the output of the following program

```
from sympy import solve
x = Symbol('x')
expr = x**2 + 5*x + 4
solve(expr, dict=True)
```

- a. $\{x: -4\}, \{x: -1\}$
- b. $\{x: -6\}, \{x: -1\}$
- c. $\{x: -1\}, \{x: -4\}$
- d. $\{x: 4\}, \{x: -1\}$

18. A *rational expression* is an algebraic expression in which the numerator and denominator are both -----

- a. Equal
- b. **Polynomials**
- c. Unequal
- d. Symmetric

19. Find the output of the following program

```
# import sympy
from sympy import *
```

```

x = symbols('x')
expr = sin(x)/x;
print("Expression : {}".format(expr))

# Use sympy.limit() method
limit_expr = limit(expr, x, 0)
print("Limit of the expression tends to 0 : {}".format(limit_expr))

```

- a. Expression : $\cos(x)/x$
Limit of the expression tends to 0 : 2
- b. Expression : $\tan(x)/x$
Limit of the expression tends to 0 : 3
- c. Expression : $\sin(x)/x$
Limit of the expression tends to 1 : 0
- d. **Expression : $\sin(x)/x$**
Limit of the expression tends to 0 : 1

20. ----- method will simplify mathematical expression using trigonometric identities.

- a. **sympy.trigsimp()**
- b. sympy.series()
- c. sympy.lambda()
- d. sympy.sim()

Which of the following programming paradigms allow us to write programs and know they are correct before running them?

- a) Automata based Programming Paradigm
- b) Logical Programming Paradigm
- c) **Dependent type Programming Paradigm**
- d) Imperative Programming Paradigm

2) Which of the following is false regarding dependent types?

- a) They allow us to write programs and know they are correct before running them.
- b) **They allow us to write programs and know they are correct only after running them.**
- c) You can specify types that can check the value of your variables at compile time.
- d) Its definition depends on a value.

3) Which of the notations is true for “P(x) is true for all values of x in the universe of discourse”?

- a) $x\forall P(x)$
- b) $\forall P(x)x$
- c) $\exists xP(x)$
- d) **$\forall xP(x)$**

4) Which of the following is false about quantifiers?

- a) Notation \forall is used for Universal quantifier.
- b) Notation \exists is used for Existential quantifier.
- c) “All men drink tea” is an example of Universal Quantifier
- d) **“All men drink coffee” is an example of Existential Quantifier.**

5) Let P(x) be the predicate “x must take a discrete mathematics course” and let Q(x) be the predicate “x is a computer science student”.

Which of the following statements is correct for “Everybody must take a discrete mathematics course or be a computer science student”?

- a) **$\forall x(Q(x) \vee P(x))$**
- b) $\forall x(Q(x)) \vee \forall x(P(x))$
- c) $\forall x(Q(x) \parallel P(x))$
- d) $\forall x(Q(x) \rightarrow P(x))$

6) Which of the following is correct for predicate “All men drink coffee”?

- a) $\forall x \text{men}(x) \rightarrow (x, \text{coffee})$
- b) $\text{drink}(x, \text{coffee}) \rightarrow \forall x \text{men}(x)$

- c) $\forall x \text{ men}(x) \rightarrow \text{drink}(x, \text{coffee})$
d) $\forall \text{men}(x) \rightarrow \text{drink}(x, \text{coffee})$
- 7) Which of the following is correct for predicate "Some boys play football"?
a) $\exists x \text{ boys}(x) \rightarrow \text{play}(x, \text{football})$
b) $\exists \text{ boys}(x) \rightarrow \text{play}(x, \text{football})$
c) $\forall x \text{ boys}(x) \rightarrow \text{play}(x, \text{football})$
d) $\exists x \forall \text{boys}(x) \rightarrow \text{play}(x, \text{football})$
- 8) Dependent Type is used to encode _____ like "for all" and "there exists".
a. **Logic Quantifiers**
b. Analysing Quantifiers
c. Dependent Quantifiers
d. None of the above
- 9) "There exists x in the universe of discourse such that P(X) is true" is which Quantifiers statement?
a. Logical
b. Universal
c. **Existential**
d. None of these
- 10) What is the way to determine if a given function is dependant type
a. Result is independent of the argument
b. Result depends upon the usage in the program
c. **Result depends on the Value of its argument**
d. Result depends on available resources
- 11) Notation for an Existential Quantifier:
a. $\forall x P(x)$
b. $\Sigma P(x)$
c. $\emptyset x P(x)$
d. **$\exists x P(x)$**
- 12) Representation of the following statement:
Every Clock has quartz
a. $\emptyset x \text{clock}(x) \rightarrow \text{quartz}(x)$
b. $\exists x \text{clock}(x) \rightarrow \text{quartz}(x)$
c. **$\forall x \text{clock}(x) \rightarrow \text{quartz}(x)$**
d. none of the above
- 13) Representation of the following statement:
Some leaves are Red

a. $\exists x \text{leaves}(x) \rightarrow \text{red}(x)$

b. $\forall x \text{leaves}(x) \rightarrow \text{red}(x)$

c. $\Sigma \text{leaves}(x) \rightarrow \text{red}(x)$

d. none of the above

13. A function has dependent type if the _____ of a function's result depends on the _____ of its Argument

a. value and type

b. type and value

c. type and type

d. value and value

14. Dependent type paradigm used to encode logic's quantifiers like _____ and _____

a. for one, there may exists

b. for all, there always

c. for all, there exists

d. for specific, there exists

15. Choose the correct one with respect to typing and typing-extensions library in python dependent type programming.

a. typing is not builtin python module where all possible types are defined.

typing_extensions is an official package for new types in the future releases of python

b. typing is a builtin python module where all possible types are defined.

typing_extensions is an official package for new types in the future releases of python

c. typing is a builtin python module where all possible types are defined.

typing_extensions is not an official package for new types in the future releases of python

d. typing is not a builtin python module where all possible types are defined.

typing_extensions is not an official package for new types in the future releases of python

16. Predict the output of the below mentioned python code without dependent type syntax:

```
def make_hamburger(meat, number_of_meats):
    return ["bread"] + [meat] * number_of_meats + ["bread"]
print(make_hamburger("ground beef", 2))
```

a. ['bread', 'ground beef', 2, 'bread']

b. ['bread', 'ground beef', 'ground beef', 'bread']

c. *TypeError: cannot concatenate 'str' and 'int' objects*

d. `['bread', 'ground beef', '2bread']`

17. Predict the output of the below mentioned python code with dependent type syntax:

```
from typing import List

def greet_all(names: List[str]) -> None:
    for name in names:
        print('Hello ' + name)

names = ["Alice", "Bob", "Charlie"]
ages = [10, 20, 30]

greet_all(names)
greet_all(ages)

a. greet_all(names) # Ok!
    greet_all(ages) # Ok!

b. a.greet_all(names) # Ok!
    greet_all(ages) # Error due to incompatible types

c. a.greet_all(names) # Error due to incompatible types
    greet_all(ages) # Ok!

d. a.greet_all(names) # Error due to incompatible types
    greet_all(ages) # Error due to incompatible types
```

18 Let x be a variable which refers to Universe of Disclosure such as x1,x2....xn then

,how to represent this statement using quantifiers “ All Man working in Industry”

- a) $\forall x \text{ man}(x) \rightarrow \text{work}(x, \text{Industry})$ b) $\forall x \text{ man}(x) \rightarrow \text{work}(\text{industry})$.
- a) $\forall x \rightarrow \text{work}(x, \text{industry})$ d) $\forall x \text{ man}(x) \rightarrow \text{Industry}(\text{work})$

19 How do you represent the statement “Every man respects his parent.”

- a) $\forall x \text{ woman}(x) \rightarrow \text{respects}(x, \text{parent})$ b) $\forall x \text{ man}(x) \rightarrow \text{respects}(x, \text{parent})$
- c) $\forall x \text{ man} \rightarrow \text{respects}(x, \text{parent})$ d) $\forall x \text{ man}(x) \rightarrow \text{respects}(\text{parent})$.

20 How do you represent the statement “Some boys play cricket “

- a) $\exists x \text{ boys}(x) \rightarrow \text{play}(\text{all})$ b) $\forall x \text{ boys}(x) \rightarrow \text{play}(x, \text{cricket})$
- c) $\exists x \text{ boys}(x) \rightarrow \text{play}(x, \text{cricket})$ d) $\exists x \wedge \forall x \text{ boys}(x) \rightarrow \text{play}(x, \text{cricket})$

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
VADAPALANI CAMPUS
DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

18CSC207J – ADVANCED PROGRAMMING PRACTICES

NETWORK PROGRAMMING PARADIGM

1. _____ programming is a way of connecting two nodes on a network to communicate with each other.
 - i. Logic
 - ii. **Socket**
 - iii. Functional
 - iv. Symbolic

2. Which method used by the server to initiate the connection with the client?
 - i. Listen()
 - ii. Close()
 - iii. Bind()
 - iv. **Accept()**

3. _____ is a socket through which data can be transmitted continuously.
 - i. Datagram Socket
 - ii. **Stream Socket**
 - iii. Raw Socket
 - iv. Binary Socket

4. _____ paradigm deals with client – server communication
 - i. Dependent type paradigm
 - ii. Parallel programming paradigm
 - iii. Network paradigm
 - iv. Concurrent programming paradigm

5. _____ protocol facilitates sending of datagrams in an unreliable manner.

- i. TCP
 - ii. UDP**
 - iii. HTTP
 - iv. FTP
- 6. Which among methods are not server socket?
 - i. connect()**
 - ii. bind()
 - iii. listen()
 - iv. accept()
- 7. In UDP, Which among methods are used to receive messages at endpoint
 - i. sock_object.recv()
 - ii. sock_object.send()
 - iii. sock_object.recvfrom()**
 - iv. sock_object.sendto()
- 8. In TCP, Which among methods are used to send messages from endpoint
 - i. sock_object.recv()
 - ii. sock_object.send()**
 - iii. sock_object.recvfrom()
 - iv. sock_object.sendto()
- 9. The protocol which defines IPv4 is
 - i. AF_UNIX
 - ii. SOCK_STREAM
 - iii. AF_INET**
 - iv. SOCK_DGRAM
- 10. The correct order of methods used in server socket is
 - i. Socket(), Bind(), listen(), accept()**
 - ii. Socket(), listen(), bind(), accept()
 - iii. Socket(), accept(), bind(), listen()
 - iv. Socket(), bind(), accept(), listen()

11. _____ is a type of network socket which provides connection less point for sending and receiving packets.

- i. **Datagram Socket**
- ii. Stream Socket
- iii. Raw Socket
- iv. Binary Socket

12. _____ do not use any transport protocol but data is directly transmitted over IP protocol

- i. Datagram Socket
- ii. Stream Socket
- iii. **Raw Socket**
- iv. Binary Socket

13. The _____ is a physical path over which the message travels.

- i. Protocol
- ii. **Medium**
- iii. Path
- iv. Route

14. A pair (host, port) is used for the _____ address family.

- i. AF_NETLINK
- ii. AF_INET6
- iii. **AF_INET**
- iv. AF_ALG

15. Use _____ to make the socket to visible to the outside world.

- i. socket.listen()
- ii. socket.visible()
- iii. socket.socket()
- iv. **Socket.gethostname()**

16. In which mode socket is created in default.

- i. **Blocking mode**
- ii. Non-Blocking Mode
- iii. Timeout mode
- iv. Accept mode

17. Which method is recommended to be called before calling connect() method?

- i. getdefaulttimeout()

- ii. **settimeout()**
- iii. **getaddrinfo()**
- iv. no such method

18. Which exception is raised for address related errors by getaddrinfo()?

- i. gaoerror
- ii. gsierror
- iii. **gaierror**
- iv. gdeerror

19. _____ protocol facilitates sending of datagrams in an reliable manner.

- i. **TCP**
- ii. UDP
- iii. HTTP
- iv. FTP

20. To create a socket, which function among the following is available in python socket module?

- i. socket.create()
- ii. socket.initialize()
- iii. **socket.socket()**
- iv. socket.build()

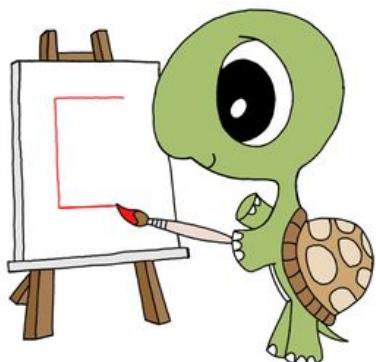


Event-Driven Programming

Using Turtle

Ref:

1. How to Think Like a Computer Scientist: Learning with Python
3
2. Review of Programming Paradigms Throughout the History
With a Suggestion Toward a Future Approach



Event- Driven Programming

- A dominant paradigm used in **Graphical User Interface (GUI)**
- Flow of the programs in GUI is determined by user actions also known as events
 - Mouse clicks, key presses, sensor outputs and so on
- Event driven applications use a loop to detect events
- Occurrence of an event triggers a function called as Event Handlers
- Event-driven programs can be easily developed in all object-oriented languages and Visual languages comparatively.
 - Eg.: Visual Basic, Visual C++, Java, VB.Net

How Event-Driven Programming works?

- **Event Scheduler** is central to event-driven programming
- Scheduler receives an event and passes the event to its respective handler
- Each event has an Unique ID and additional information such as, x and y coordinates of mouse pointer in a mouse event or state of shift key in key press event
- **How Events are generated?**
 - Actions performed by user during the execution of a program like mouse click, pressing a button, selecting a choice button or a radio button
 - Messages generated by peripheral device, System hardware
 - Completion of file download generates an event

Event Handlers

- A block of code that executes when an event occurs.
- Produces a visual response to inform/direct the user
- Changes the system's state
 - System state includes data used by the system and the state of the user interface-on-screen object has the focus
 - Sometimes event handler calls another event handler.
- Event handlers are linked to the on screen object to which it should listen-called as binding

Event-Driven Programming

- Python is used to demonstrate Event-driven programs
- Libraries that support Event driven programming in Python are
 - Turtle
 - Tkinter
 - PyQt5
 - WxWidgets

Turtle

- Turtle is a library included in Python
- Turtle is used to draw shapes and patterns and handle events
- Turtle module is imported

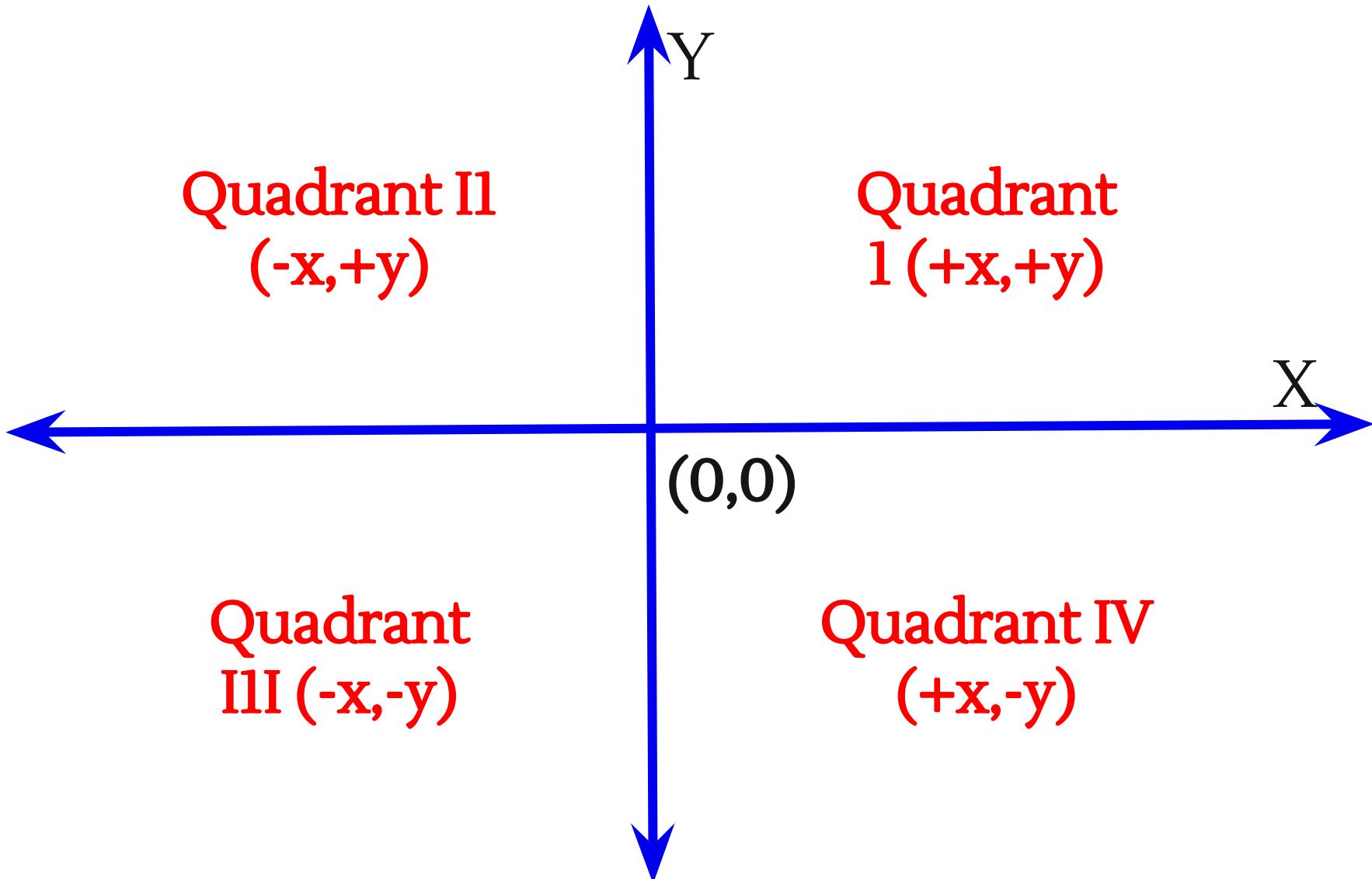
```
import turtle * or from turtle import *
```

- Following command opens a window called as screen and a black triangle in the middle called turtle

```
s = turtle.getscreen()
```

- Screen acts as a canvas and turtle acts as a pen
- Turtle has specific characteristics like, size, colour, speed and direction in which the turtle can move

Turtle Screen



Turtle commands

- Movement
 - `forward(x)/ fd(x)`-moves turtle in forward direction by x units
 - `backward(x)/bk(x)/back(x)`-moves turtle in backward by x units
 - `left(d)/lt(d)`-turns the turtle left by d degrees
 - `right(d)/rt(d)`-turns the turtle right by d degrees

Other commands

- goto() | setpos() |
setposition()
- setx()
- sety()
- setheading() | seth()
- home()
- circle()
- dot()
- stamp()
- clearstamp()
- clearstamps()
- undo()
- speed()
- And many more such
commands are available

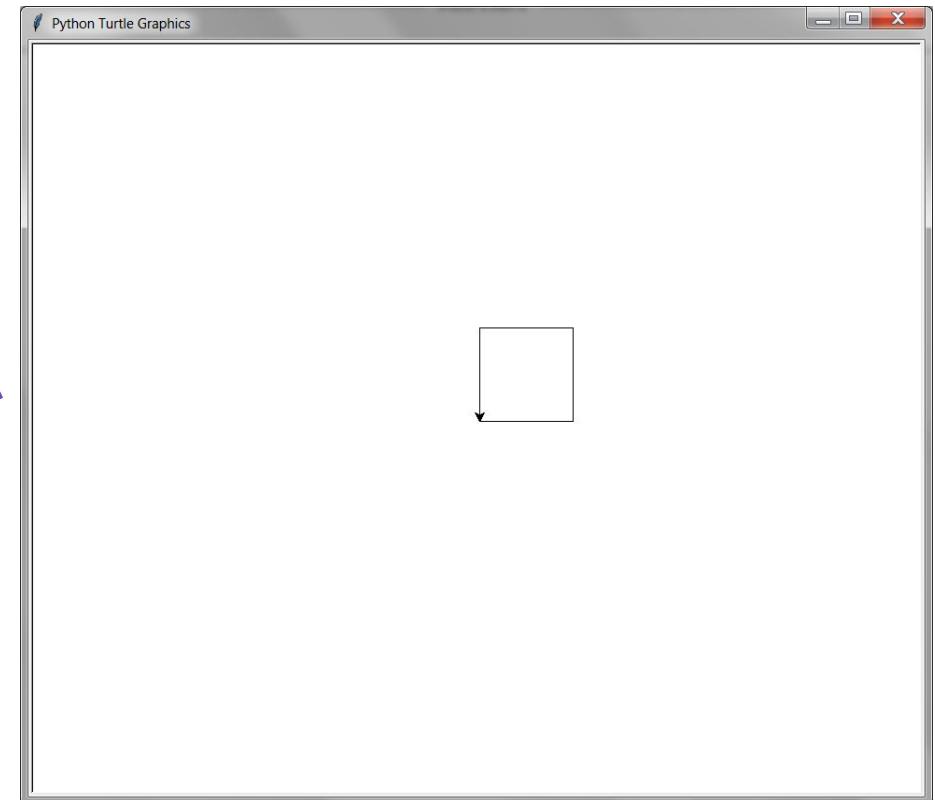
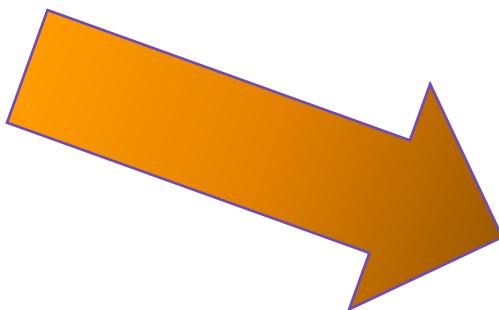
Event handling commands

- onclick()
- onrelease()
- ondrag()
- listen()
- onkey() | onkeyrelease()
- onkeypress()
- onclick() | onscreenclick()
- ontimer()
- mainloop() | done()

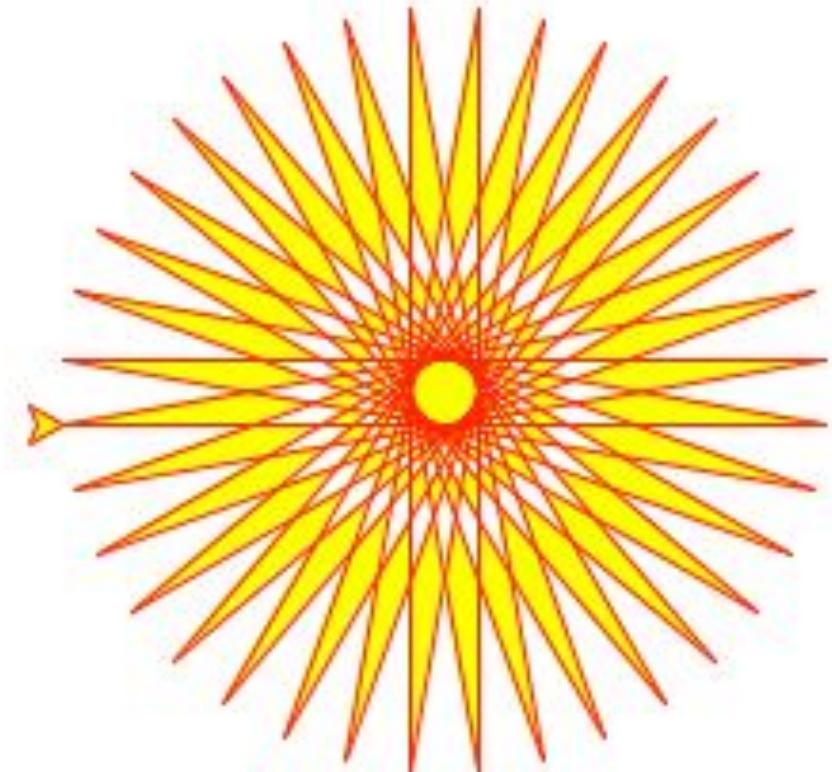
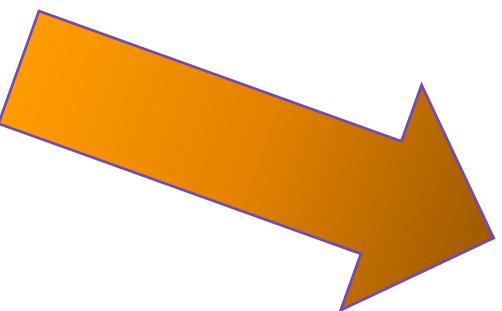
Simple programs using turtle

```
from turtle import *
```

```
forward(100)  
left(90)  
forward(100)  
left(90)  
forward(100)  
left(90)  
forward(100)
```



```
from turtle import *
color('red', 'yellow')
begin_fill()
while True:
    forward(200)
    left(170)
    if abs(pos()) < 1:
        break
end_fill()
done()
```

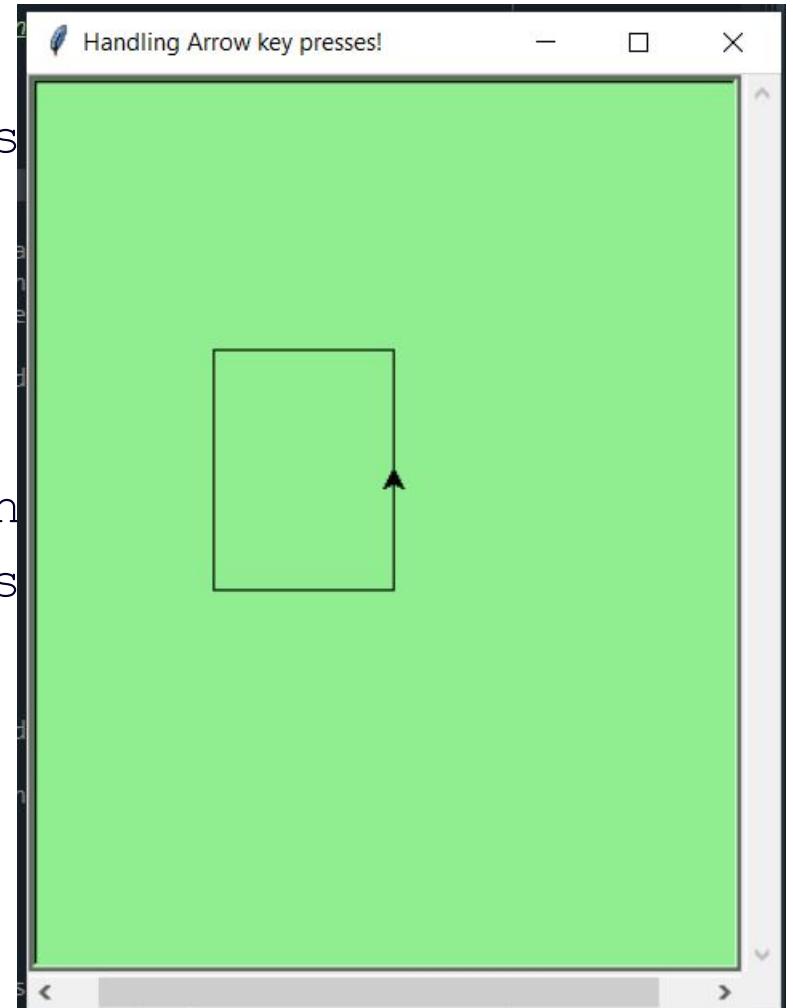
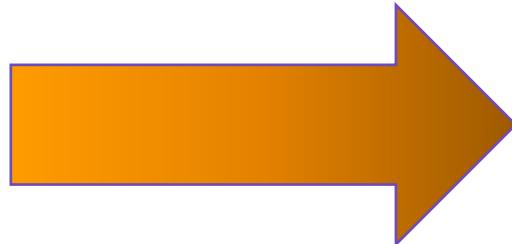


Keypress event

```
import turtle  
turtle.setup(400,500,10,20)      # Determine the window size  
wn = turtle.Screen()            # Get a reference to the window  
wn.title("Handling keypresses!") # Change the window title  
wn.bgcolor("lightgreen")        # Set the background color  
tess = turtle.Turtle()          # Create our favorite turtle  
# The next four functions are our "event handlers".  
def h1():  
    tess.forward(30)  
def h2():  
    tess.left(45)  
def h3():  
    tess.right(45)
```

Contd...

```
def h4():
    wn.bye()          # Close down the turtle window
# These lines "wire up" keypresses to the handlers
wn.onkey(h1, "Up")
wn.onkey(h2, "Left")
wn.onkey(h3, "Right")
wn.onkey(h4, "q")
# Now we need to tell the window to start listening
# If any of the keys that we're monitoring is pressed
# handler will be called.
wn.listen()
wn.mainloop()
```



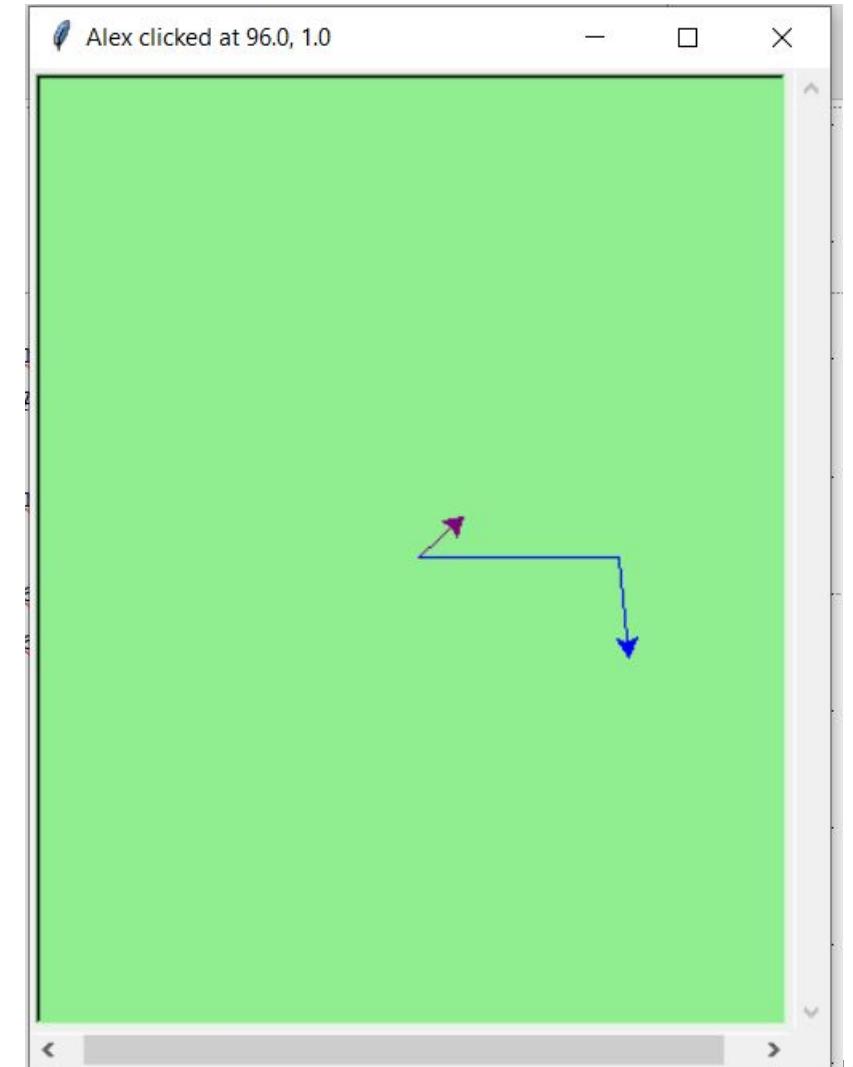
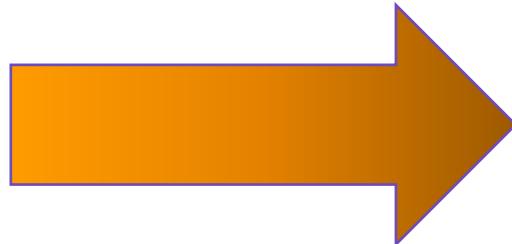
Mouse Event

```
import turtle
turtle.setup(400,500,10,20)          # Determine the window size
wn = turtle.Screen()                 # Get a reference to the window
wn.title("Handling mouse clicks!")  # Change the window title
wn.bgcolor("lightgreen")             # Set the background color
tess = turtle.Turtle()               # Create two turtles
tess.color("purple")
alex = turtle.Turtle()              # Move them apart
alex.color("blue")
alex.forward(100)

def handler_for_tess(x, y):
    wn.title("Tess clicked at {0}, {1}".format(x, y))
    tess.left(42)
    tess.forward(30)
```

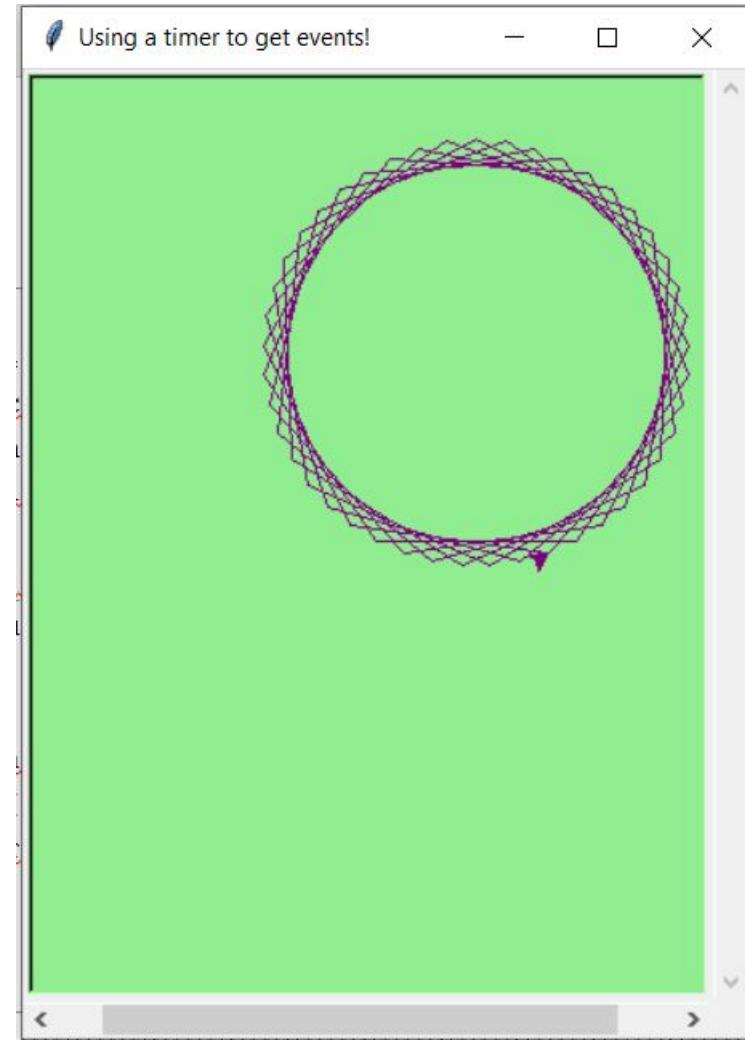
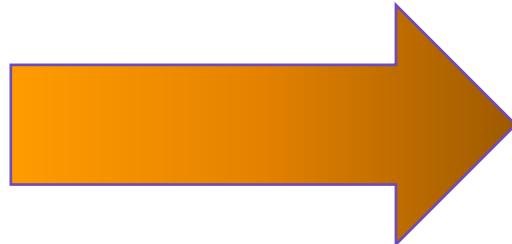
Contd...

```
def handler_for_alex(x, y):  
    wn.title("Alex clicked at {0}, {1}".format(x, y))  
    alex.right(84)  
    alex.forward(50)  
  
tess.onclick(handler_for_tess)  
alex.onclick(handler_for_alex)  
  
wn.mainloop()
```



Automatic event from Timer

```
import turtle  
turtle.setup(400,500)  
wn = turtle.Screen()  
wn.title("Using a timer to get events!")  
wn.bgcolor("lightgreen")  
  
tess = turtle.Turtle()  
tess.color("purple")  
  
def h1():  
    tess.forward(100)  
    tess.left(56)  
    wn.ontimer(h1, 60)  
  
h1()  
wn.mainloop()
```



References

- https://github.com/asweigart/simple-turtle-tutorial-for-python/blob/master/simple_turtle_tutorial.md
- <https://docs.python.org/3/library/turtle.html>
- <https://openbookproject.net/thinkcs/python/english3e/events.html#:~:text=The%20turtle%20module%20in%20Python,when%20its%20time%20is%20up.&text=On%20line%202016%20the%20timer,~and%20tess%20springs%20into%20action.>

18CSC207J – Advanced Programming Practice

GUI & Event Handling Programming Paradigm

Event Driven Programming Paradigm

- Event-driven programming is a programming paradigm in which the flow of program execution is determined by events - for example a user action such as a mouse click, key press, or a message from the operating system or another program.
- An event-driven application is designed to detect events as they occur, and then deal with them using an appropriate event-handling procedure.
- In a typical modern event-driven program, there is no discernible flow of control. The main routine is an event-loop that waits for an event to occur, and then invokes the appropriate event-handling routine.
- Event callback is a function that is invoked when something significant happens like when click event is performed by user or the result of database query is available.

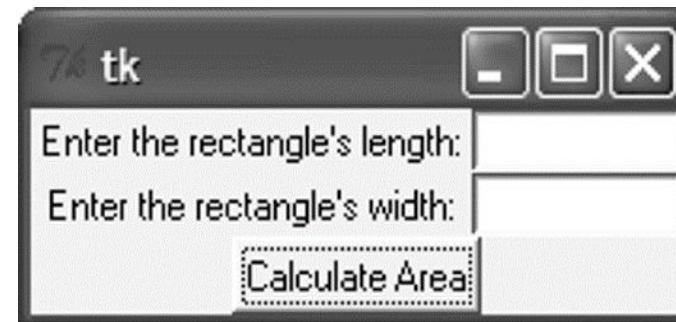
Event Handlers: Event handlers is a type of function or method that run a specific action when a specific event is triggered. For example, it could be a button that when user click it, it will display a message, and it will close the message when user click the button again, this is an event handler.

Trigger Functions: Trigger functions in event-driven programming are a functions that decide what code to run when there are a specific event occurs, which are used to select which event handler to use for the event when there is specific event occurred.

Events: Events include mouse, keyboard and user interface, which events need to be triggered in the program in order to happen, that mean user have to interacts with an object in the program, for example, click a button by a mouse, use keyboard to select a button and etc.

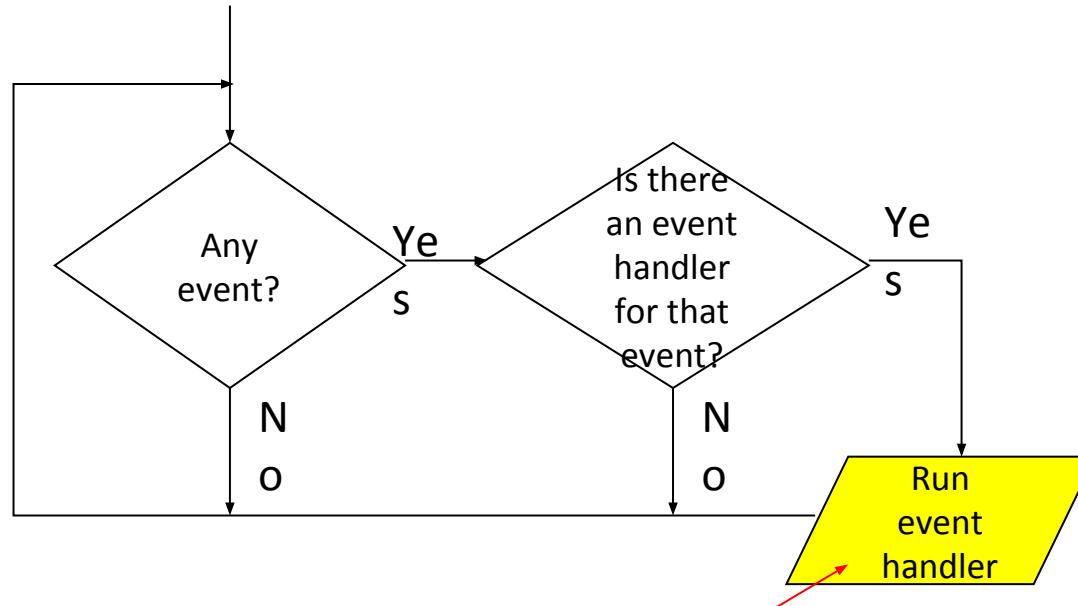
Introduction

- A graphical user interface allows the user to interact with the operating system and other programs using graphical elements such as icons, buttons, and dialog boxes.
- GUIs popularized the use of the mouse.
- GUIs allow the user to point at graphical elements and click the mouse button to activate them.
- GUI Programs Are Event-Driven
- User determines the order in which things happen
- GUI programs respond to the actions of the user, thus they are event driven.
- The tkinter module is a wrapper around tk, which is a wrapper around tcl, which is what is used to create windows and graphical user interfaces.



Introduction

- A major task that a GUI designer needs to do is to determine what will happen when a GUI is invoked
- Every GUI component may generate different kinds of “events” when a user makes access to it using his mouse or keyboard
- E.g. if a user moves his mouse on top of a button, an event of that button will be generated to the Windows system
- E.g. if the user further clicks, then another event of that button will be generated (actually it is the click event)
- For any event generated, the system will first check if there is an event handler, which defines the action for that event
- For a GUI designer, he needs to develop the event handler to determine the action that he wants Windows to take for that event.



GUI Using Python

- Tkinter: Tkinter is the Python interface to the Tk GUI toolkit shipped with Python.
- wxPython: This is an open-source Python interface for wxWindows
- PyQt –This is also a Python interface for a popular cross-platform Qt GUI library.
- JPython: JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine

Tkinter Programming

- Tkinter is the standard GUI library for Python.
- Creating a GUI application using Tkinter

Steps

- Import the Tkinter module.

Import tkinter as tk

- Create the GUI application main window.

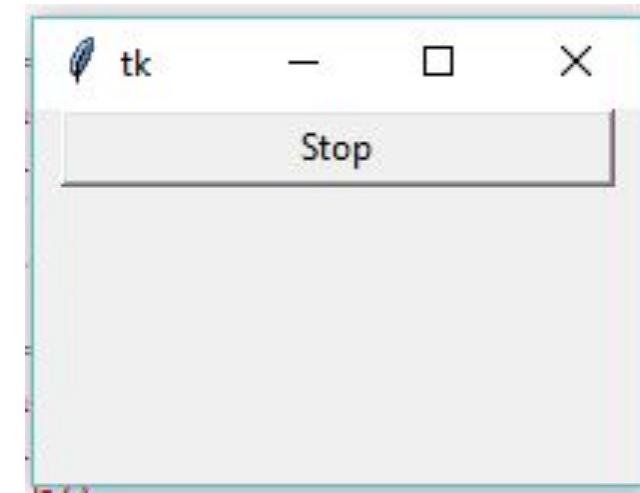
root = tk.Tk()

- Add one or more of the above-mentioned widgets to the GUI application.

*button = tk.Button(root, text='Stop', width=25, command=root.destroy)
button.pack()*

- Enter the main event loop to take action against each event triggered by the user.

root.mainloop()

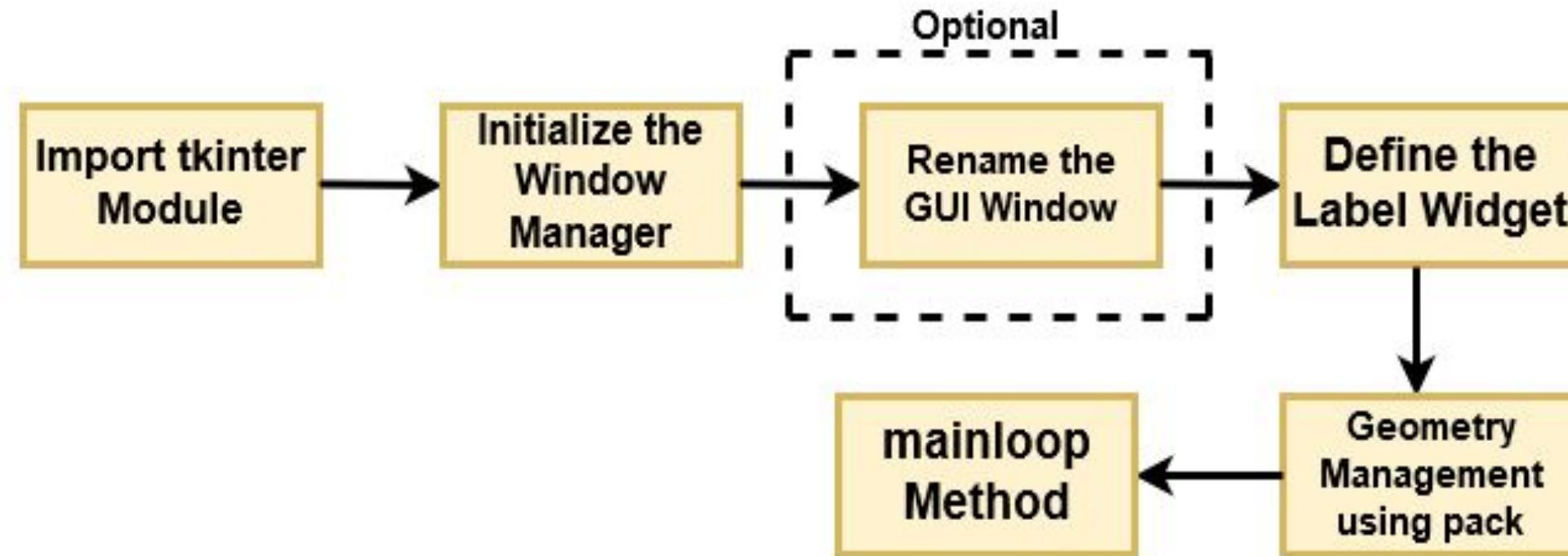


Tkinter widgets

- Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

Widget	Description
Label	Used to contain text or images
Button	Similar to a Label but provides additional functionality for mouse overs, presses, and releases as well as keyboard activity/events
Canvas	Provides ability to draw shapes (lines, ovals, polygons, rectangles); can contain images or bitmaps
Radiobutton	Set of buttons of which only one can be "pressed" (similar to HTML radio input)
Checkbutton	Set of boxes of which any number can be "checked" (similar to HTML checkbox input)
Entry	Single-line text field with which to collect keyboard input (similar to HTML text input)
Frame	Pure container for other widgets
Listbox	Presents user list of choices to pick from
Menu	Actual list of choices "hanging" from a Menubutton that the user can choose from
Menubutton	Provides infrastructure to contain menus (pulldown, cascading, etc.)
Message	Similar to a Label, but displays multi-line text
Scale	Linear "slider" widget providing an exact value at current setting; with defined starting and ending values
Text	Multi-line text field with which to collect (or display) text from user (similar to HTML TextArea)
Scrollbar	Provides scrolling functionality to supporting widgets, i.e., Text, Canvas, Listbox, and Entry
Toplevel	Similar to a Frame, but provides a separate window container

Operation Using Tkinter Widget



Geometry Managers

- The pack() Method – This geometry manager organizes widgets in blocks before placing them in the parent widget.

`widget.pack(pack_options)`

options

- expand – When set to true, widget expands to fill any space not otherwise used in widget's parent.
 - fill – Determines whether widget fills any extra space allocated to it by the packer, or keeps its own minimal dimensions: NONE (default), X (fill only horizontally), Y (fill only vertically), or BOTH (fill both horizontally and vertically).
 - side – Determines which side of the parent widget packs against: TOP (default), BOTTOM, LEFT, or RIGHT.
- The grid() Method – This geometry manager organizes widgets in a table-like structure in the parent widget.

`widget.grid(grid_options)`

options –

- Column/row – The column or row to put widget in; default 0 (leftmost column).
- Columnspan, rowsapn – How many columns or rows to widget occupies; default 1.
- ipadx, ipady – How many pixels to pad widget, horizontally and vertically, inside widget's borders.
- padx, pady – How many pixels to pad widget, horizontally and vertically, outside v's borders.

Geometry Managers

- The place() Method – This geometry manager organizes widgets by placing them in a specific position in the parent widget.

widget.place(place_options)

options –

- anchor – The exact spot of widget other options refer to: may be N, E, S, W, NE, NW, SE, or SW, compass directions indicating the corners and sides of widget; default is NW (the upper left corner of widget)
- bordermode – INSIDE (the default) to indicate that other options refer to the parent's inside (ignoring the parent's border); OUTSIDE otherwise.
- height, width – Height and width in pixels.
- relheight, relwidth – Height and width as a float between 0.0 and 1.0, as a fraction of the height and width of the parent widget.
- relx, rely – Horizontal and vertical offset as a float between 0.0 and 1.0, as a fraction of the height and width of the parent widget.
- x, y – Horizontal and vertical offset in pixels.

Common Widget Properties

Common attributes such as sizes, colors and fonts are specified.

- Dimensions
- Colors
- Fonts
- Anchors
- Relief styles
- Bitmaps
- Cursors

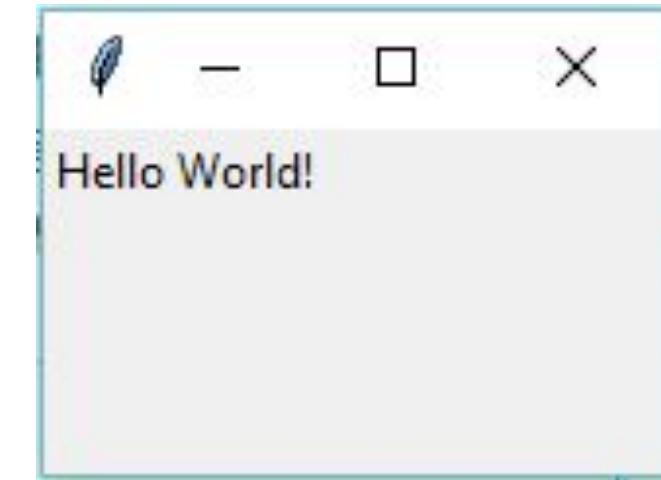
Label Widgets

- A label is a widget that displays text or images, typically that the user will just view but not otherwise interact with. Labels are used for such things as identifying controls or other parts of the user interface, providing textual feedback or results, etc.
- Syntax

tk.Label(parent, text="message")

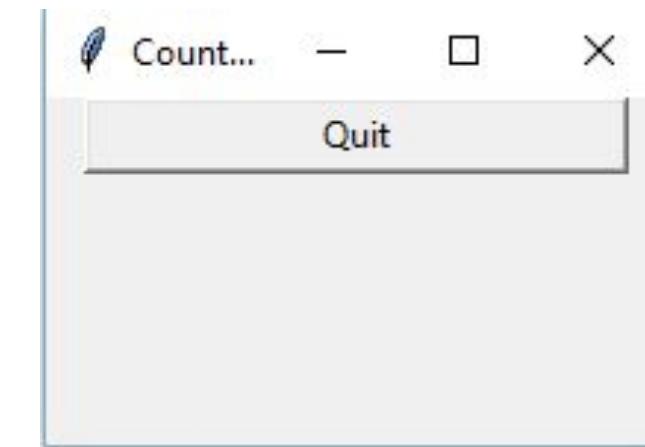
Example:

```
import tkinter as tk  
  
root = tk.Tk()  
  
label = tk.Label(root, text='Hello World!')  
  
label.grid()  
  
root.mainloop()
```



Button Widgets

```
import tkinter as tk  
  
r = tk.Tk()  
  
r.title('Counting Seconds')  
  
button = tk.Button(r, text='Stop', width=25, command=r.destroy)  
  
button.pack()  
  
r.mainloop()
```



Button Widgets

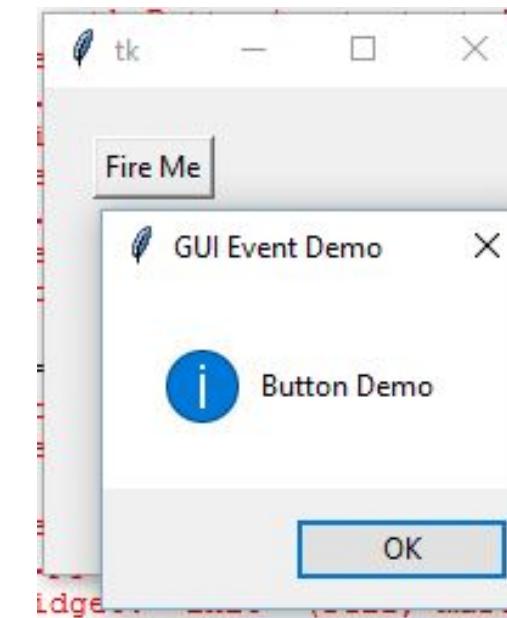
- A button, unlike a frame or label, is very much designed for the user to interact with, and in particular, press to perform some action. Like labels, they can display text or images, but also have a whole range of new options used to control their behavior.

Syntax

```
button = ttk.Button(parent, text='ClickMe', command=submitForm)
```

Example:

```
import tkinter as tk  
  
from tkinter import messagebox  
  
def hello():  
  
    msg = messagebox.showinfo( "GUI Event Demo","Button Demo")  
  
root = tk.Tk()  
  
root.geometry("200x200")  
  
b = tk.Button(root, text='Fire Me',command=hello)  
  
b.place(x=50,y=50)  
  
root.mainloop()
```



Button Widgets

- Button: To add a button in your application, this widget is used.

Syntax :

```
w=Button(master, text="caption" option=value)
```

- master is the parameter used to represent the parent window.
- activebackground: to set the background color when button is under the cursor.
- activeforeground: to set the foreground color when button is under the cursor.
- bg: to set he normal background color.
- command: to call a function.
- font: to set the font on the button label.
- image: to set the image on the button.
- width: to set the width of the button.
- height: to set the height of the button.

Entry Widgets

- An entry presents the user with a single line text field that they can use to type in a string value. These can be just about anything: their name, a city, a password, social security number, and so on.

Syntax

```
name = ttk.Entry(parent, textvariable=username)
```

Example:

```
def hello():

    msg = messagebox.showinfo( "GUI Event Demo",t.get())

root = tk.Tk()

root.geometry("200x200")

l1=tk.Label(root,text="Name:")

l1.grid(row=0)

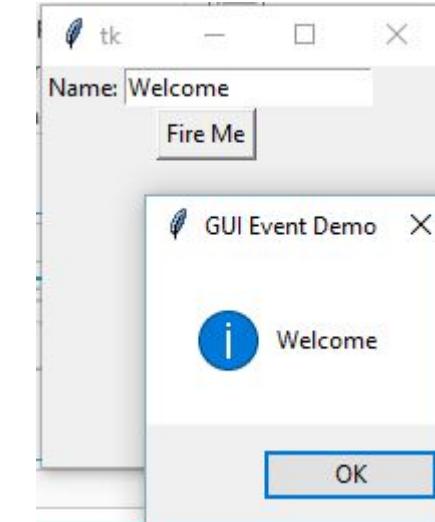
t=tk.Entry(root)

t.grid(row=0,column=1)

b = tk.Button(root, text='Fire Me',command=hello)

b.grid(row=1,columnspan=2);

root.mainloop()
```



Canvas

- The Canvas is a rectangular area intended for drawing pictures or other complex layouts. You can place graphics, text, widgets or frames on a Canvas.
- It is used to draw pictures and other complex layout like graphics, text and widgets.

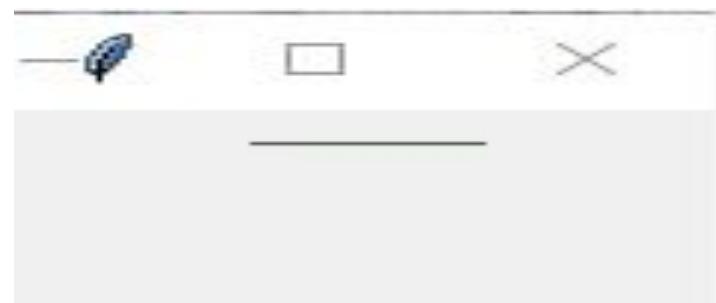
Syntax:

```
w = Canvas(master, option=value)
```

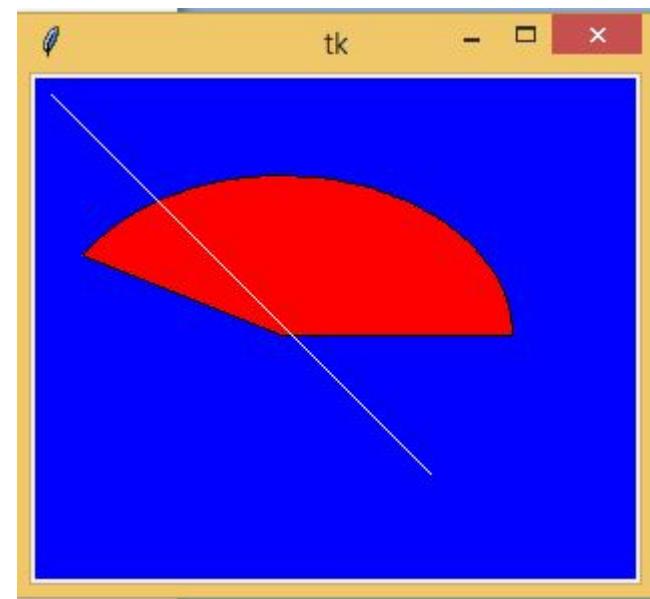
- master is the parameter used to represent the parent window.
- bd: to set the border width in pixels.
- bg: to set the normal background color.
- cursor: to set the cursor used in the canvas.
- highlightcolor: to set the color shown in the focus highlight.
- width: to set the width of the widget.
- height: to set the height of the widget.

Canvas

```
from tkinter import *\n\nmaster = Tk()\n\nw = Canvas(master, width=40, height=60)\n\nw.pack()\n\ncanvas_height=20\n\ncanvas_width=200\n\ny = int(canvas_height / 2)\n\nw.create_line(0, y, canvas_width, y )\n\nmainloop()
```



```
from tkinter import *\n\nfrom tkinter import messagebox\n\ntop = Tk()\n\nC = Canvas(top, bg = "blue", height = 250, width = 300)\n\ncoord = 10, 50, 240, 210\n\narc = C.create_arc(coord, start = 0, extent = 150, fill = "red")\n\nline = C.create_line(10,10,200,200,fill = 'white')\n\nC.pack()\n\ntop.mainloop()
```



Checkbutton

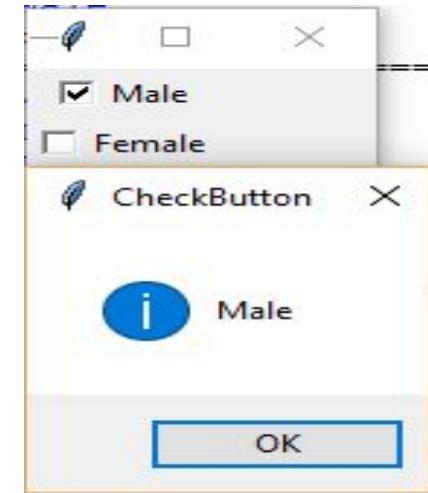
- A checkbutton is like a regular button, except that not only can the user press it, which will invoke a command callback, but it also holds a binary value of some kind (i.e. a toggle). Checkbuttons are used all the time when a user is asked to choose between, e.g. two different values for an option.

Syntax

```
w = CheckButton(master, option=value)
```

Example:

```
from tkinter import *
root= Tk()
root.title('Checkbutton Demo')
v1=IntVar()
v2=IntVar()
cb1=Checkbutton(root,text='Male', variable=v1,onvalue=1, offvalue=0, command=test)
cb1.grid(row=0)
cb2=Checkbutton(root,text='Female', variable=v2,onvalue=1, offvalue=0, command=test)
cb2.grid(row=1)
root.mainloop()
```



```
def test():
    if(v1.get()==1):
        v2.set(0)
        print("Male")
    if(v2.get()==1):
        v1.set(0)
        print("Female")
```

radiobutton

- A radiobutton lets you choose between one of a number of mutually exclusive choices; unlike a checkbutton, it is not limited to just two choices. Radiobuttons are always used together in a set and are a good option when the number of choices is fairly small

- **Syntax**

```
w = CheckButton(master, option=value)
```

Example:

```
root= Tk()  
root.geometry("200x200")  
radio=IntVar()  
rb1=Radiobutton(root,text='Red', variable=radio,width=25,value=1, command=choice)
```

```
rb1.grid(row=0)
```

```
rb2=Radiobutton(root,text='Blue', variable=radio,width=25,value=2, command=choice)
```

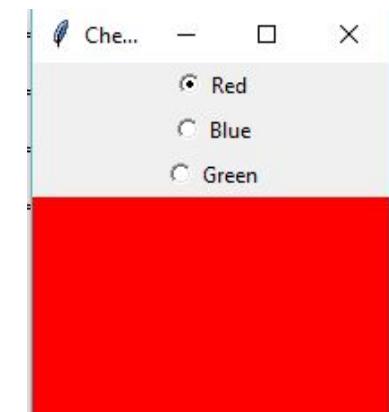
```
rb2.grid(row=1)
```

```
rb3=Radiobutton(root,text='Green', variable=radio,width=25,value=3, command=choice)
```

```
rb3.grid(row=3)
```

```
root.mainloop()
```

```
def choice():  
    if(radio.get()==1):  
        root.configure(background='red')  
    elif(radio.get()==2):  
        root.configure(background='blue')  
    elif(radio.get()==3):  
        root.configure(background='green')
```



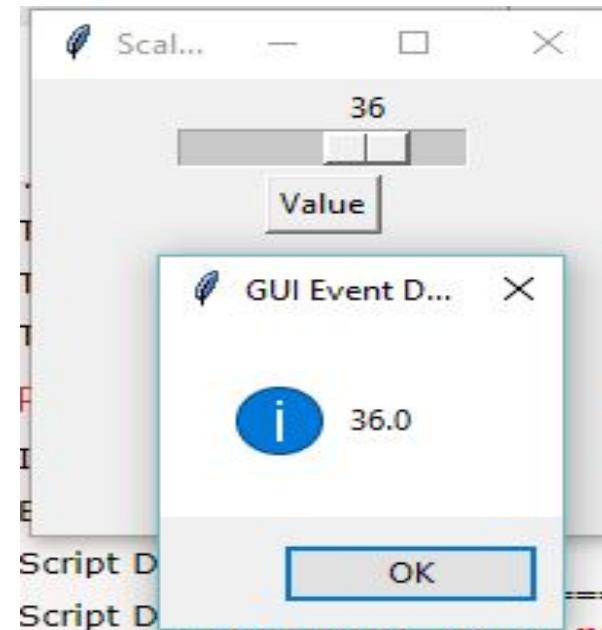
Scale

- Scale widget is used to implement the graphical slider to the python application so that the user can slide through the range of values shown on the slider and select the one among them. We can control the minimum and maximum values along with the resolution of the scale. It provides an alternative to the Entry widget when the user is forced to select only one value from the given range of values.
- **Syntax**

*w = Scale(*top, options*)*

Example:

```
from tkinter import messagebox
root= Tk()
root.title('Scale Demo')
root.geometry("200x200")
def slide():
    msg = messagebox.showinfo( "GUI Event Demo",v.get())
v = DoubleVar()
scale = Scale( root, variable = v, from_ = 1, to = 50, orient = HORIZONTAL)
scale.pack(anchor=CENTER)
btn = Button(root, text="Value", command=slide)
btn.pack(anchor=CENTER)
root.mainloop()
```



Spinbox

- The Spinbox widget is an alternative to the Entry widget. It provides the range of values to the user, out of which, the user can select the one.

Syntax

```
w = Spinbox(top, options)
```

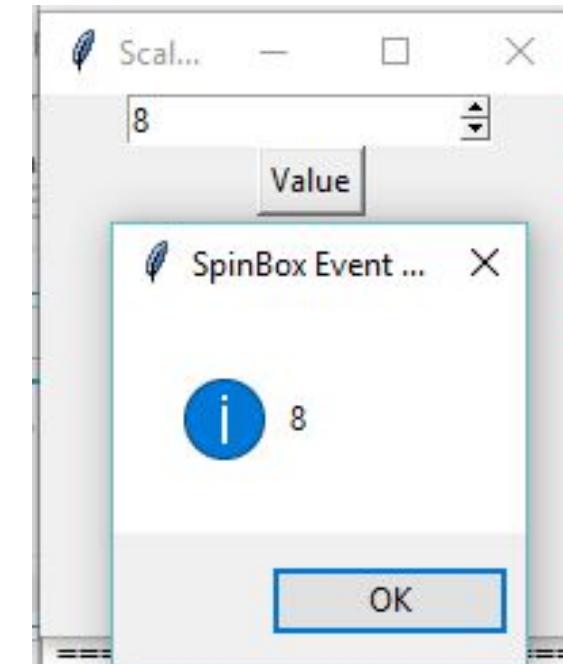
Example:

```
from tkinter import *
from tkinter import messagebox

root= Tk()
root.title('Scale Demo')
root.geometry("200x200")

def slide():
    msg = messagebox.showinfo( "SpinBox Event Demo",spin.get())

spin = Spinbox(root, from_= 0, to = 25)
spin.pack(anchor=CENTER)
btn = Button(root, text="Value", command=slide)
btn.pack(anchor=CENTER)
root.mainloop()
```



Menubutton

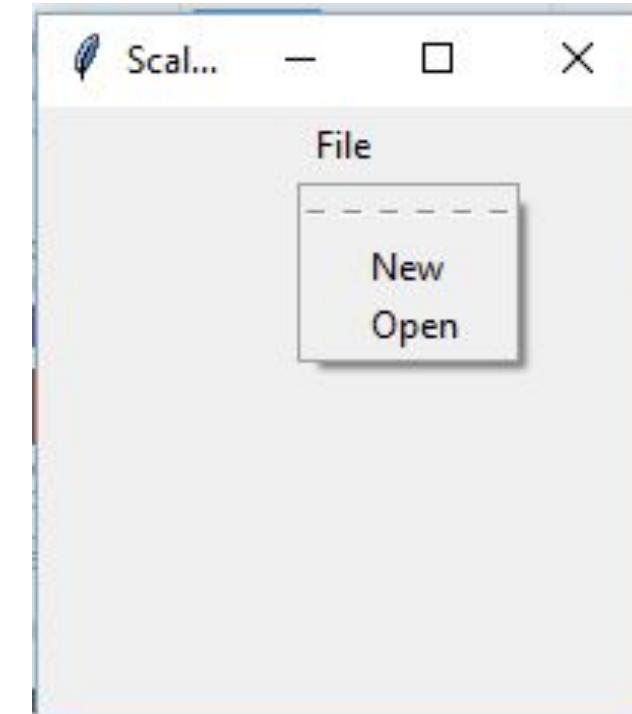
- Menubutton widget can be defined as the drop-down menu that is shown to the user all the time. It is used to provide the user a option to select the appropriate choice exist within the application.

Syntax

```
w = Menubutton(Top, options)
```

Example:

```
from tkinter import *
from tkinter import messagebox
root= Tk()
root.title('Scale Demo')
root.geometry("200x200")
menubutton = Menubutton(root, text = "File", relief = FLAT)
menubutton.grid()
menubutton.menu = Menu(menubutton)
menubutton["menu"]=menubutton.menu
menubutton.menu.add_checkbutton(label = "New", variable=IntVar(),command=)
menubutton.menu.add_checkbutton(label = "Open", variable = IntVar())
menubutton.pack()
root.mainloop()
```



Menubutton

- Menubutton widget can be defined as the drop-down menu that is shown to the user all the time. It is used to provide the user a option to select the appropriate choice exist within the application.

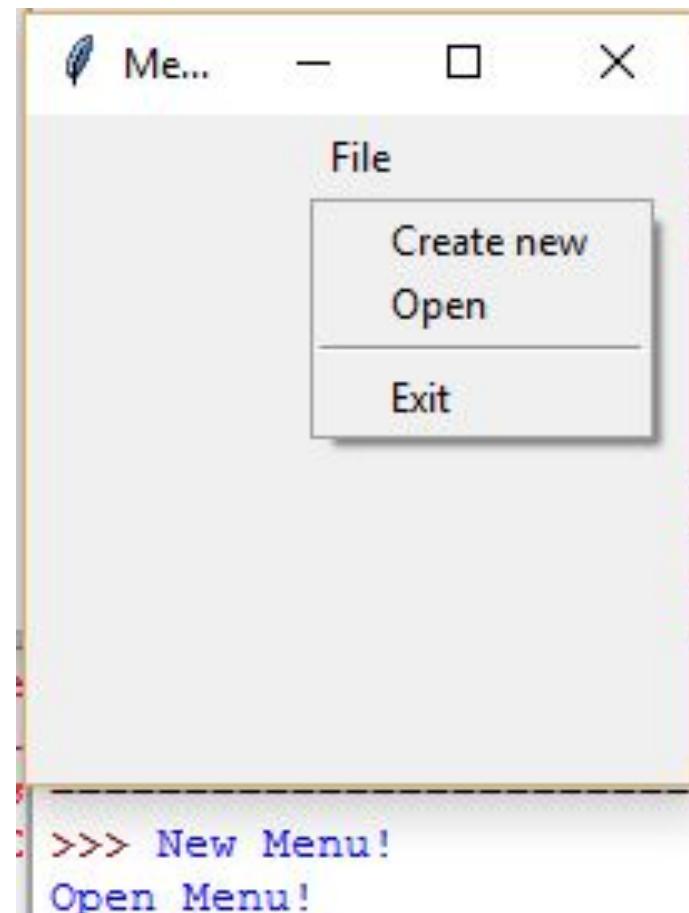
Syntax

```
w = Menubutton(Top, options)
```

Example:

```
from tkinter import *
from tkinter import messagebox
root= Tk()
root.title('Menu Demo')
root.geometry("200x200")
def new():
    print("New Menu!")
def disp():
    print("Open Menu!")

menubutton = Menubutton(root, text="File")
menubutton.grid()
menubutton.menu = Menu(menubutton, tearoff = 0)
menubutton["menu"] = menubutton.menu
menubutton.menu.add_command(label="Create
new",command=new)
menubutton.menu.add_command(label="Open",command=dis
p)
menubutton.menu.add_separator()
menubutton.menu.add_command(label="Exit",command=root.
quit)
menubutton.pack()
```

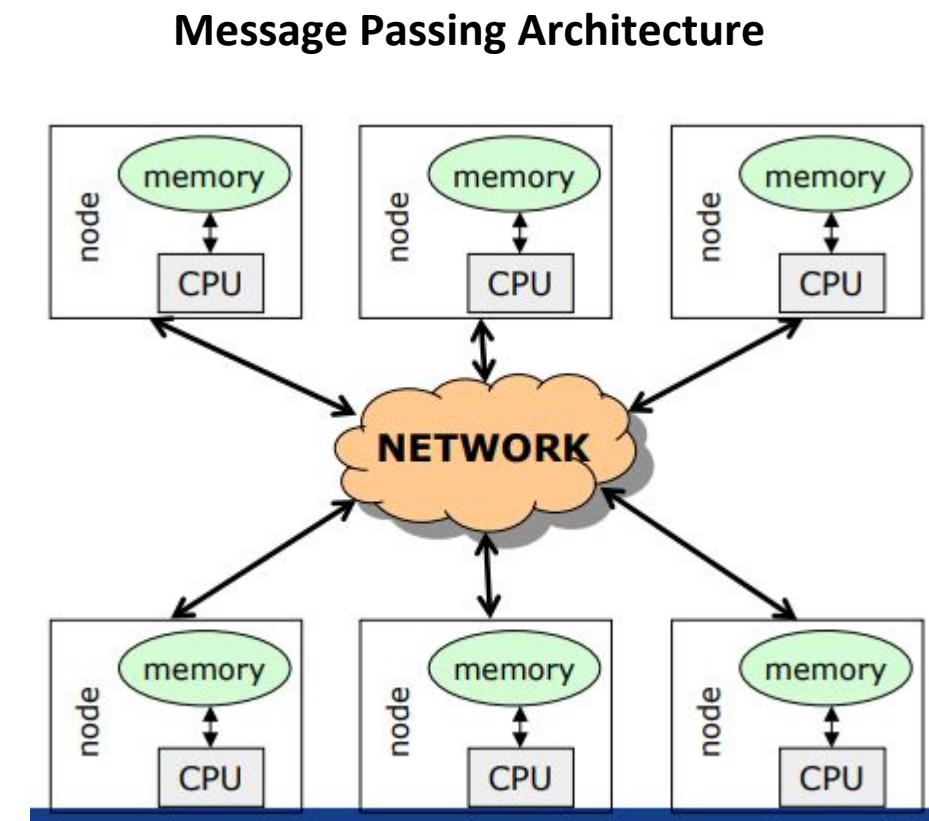
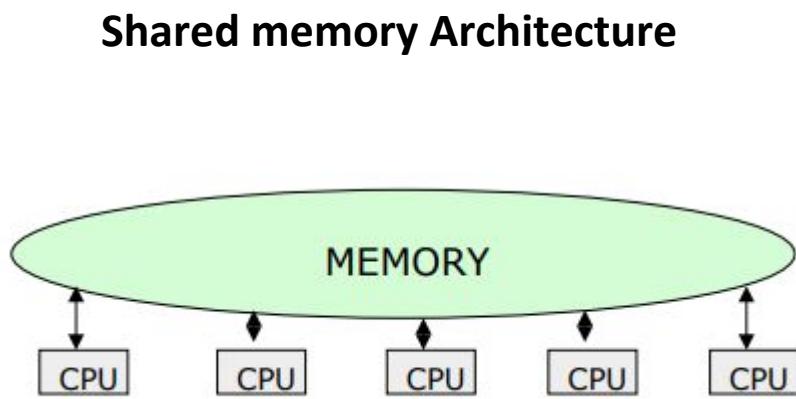


18CSC207J – Advanced Programming Practice

Parallel & Concurrent Programming Paradigm

Introduction

- A system is said to be parallel if it can support two or more actions executing simultaneously i.e., multiple actions are simultaneously executed in parallel systems.
- The evolution of parallel processing, even if slow, gave rise to a considerable variety of programming paradigms.
- Parallelism Types:
 - Explicit Parallelism
 - Implicit Parallelism



Explicit parallelism

- Explicit Parallelism is characterized by the presence of explicit constructs in the programming language, aimed at describing (to a certain degree of detail) the way in which the parallel computation will take place.
- A wide range of solutions exists within this framework. One extreme is represented by the "ancient" use of basic, low level mechanisms to deal with parallelism--like fork/join primitives, semaphores, etc--eventually added to existing programming languages. Although this allows the highest degree of flexibility (any form of parallel control can be implemented in terms of the basic low level primitives gif), it leaves the additional layer of complexity completely on the shoulders of the programmer, making his task extremely complicate.

Implicit Parallelism

- Allows programmers to write their programs without any concern about the exploitation of parallelism. Exploitation of parallelism is instead automatically performed by the compiler and/or the runtime system. In this way the parallelism is transparent to the programmer maintaining the complexity of software development at the same level of standard sequential programming.
- Extracting parallelism implicitly is not an easy task. For imperative programming languages, the complexity of the problem is almost prohibitively and allows positive results only for restricted sets of applications (e.g., applications which perform intensive operations on arrays).
- Declarative Programming languages, and in particular Functional and Logic languages, are characterized by a very high level of abstraction, allowing the programmer to focus on what the problem is and leaving implicit many details of how the problem should be solved.
- Declarative languages have opened new doors to automatic exploitation of parallelism. Their focusing on a high level description of the problem and their mathematical nature turned into positive properties for implicit exploitation of parallelism.

Methods for parallelism

There are many methods of programming parallel computers. Two of the most common are message passing and data parallel.

1. Message Passing - the user makes calls to libraries to explicitly share information between processors.
2. Data Parallel - data partitioning determines parallelism
3. Shared Memory - multiple processes sharing common memory space
4. Remote Memory Operation - set of processes in which a process can access the memory of another process without its participation
5. Threads - a single process having multiple (concurrent) execution paths
6. Combined Models - composed of two or more of the above.

Methods for parallelism

Message Passing:

- Each Processor has direct access only to its local memory
- Processors are connected via high-speed interconnect
- Data exchange is done via explicit processor-to-processor communication i.e processes communicate by sending and receiving messages : send/receive messages
- Data transfer requires cooperative operations to be performed by each process (a send operation must have matching receive)

Data Parallel:

- Each process works on a different part of the same data structure
- Processors have direct access to global memory and I/O through bus or fast switching network
- Each processor also has its own memory (cache)
- Data structures are shared in global address space
- Concurrent access to shared memory must be coordinate
- All message passing is done invisibly to the programmer

Steps in Parallelism

- Independently from the specific paradigm considered, in order to execute a program which exploits parallelism, the programming language must supply the means to:
 - Identify parallelism, by recognizing the components of the program execution that will be (potentially) performed by different processors;
 - Start and stop parallel executions;
 - Coordinate the parallel executions (e.g., specify and implement interactions between concurrent components).

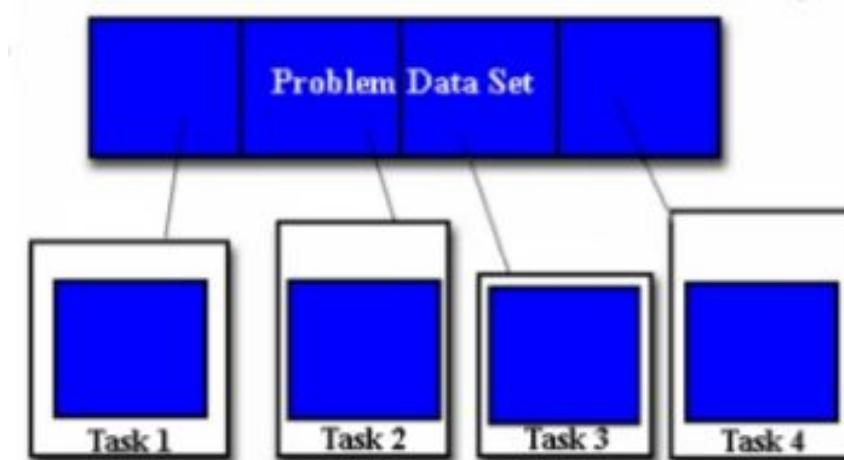
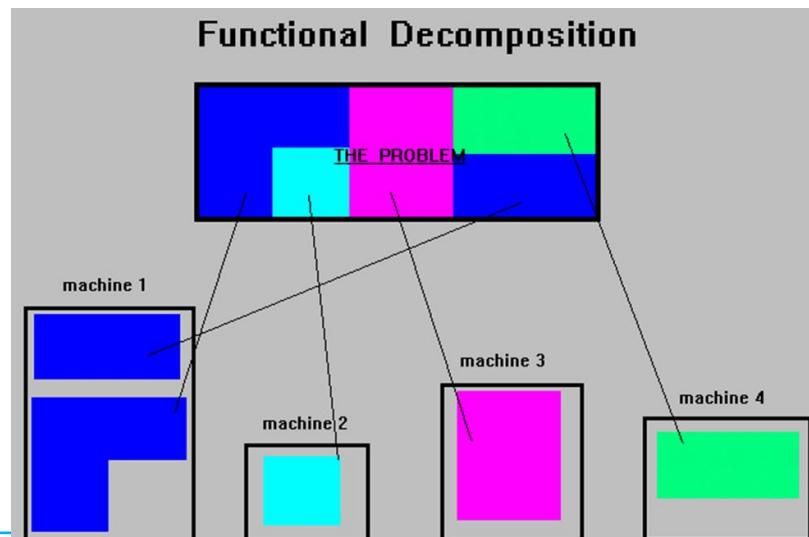
Ways for Parallelism

Functional Decomposition (Functional Parallelism)

- Decomposing the problem into different tasks which can be distributed to multiple processors for simultaneous execution
- Good to use when there is no static structure or fixed determination of number of calculations to be performed

Domain Decomposition (Data Parallelism)

- Partitioning the problem's data domain and distributing portions to multiple processors for simultaneous execution
- Good to use for problems where:
- data is static (factoring and solving large matrix or finite difference calculations)
- dynamic data structure tied to single entity where entity can be subsetted (large multi-body problems)
- domain is fixed but computation within various regions of the domain is dynamic (fluid vortices models)



Parallel Programming Paradigm

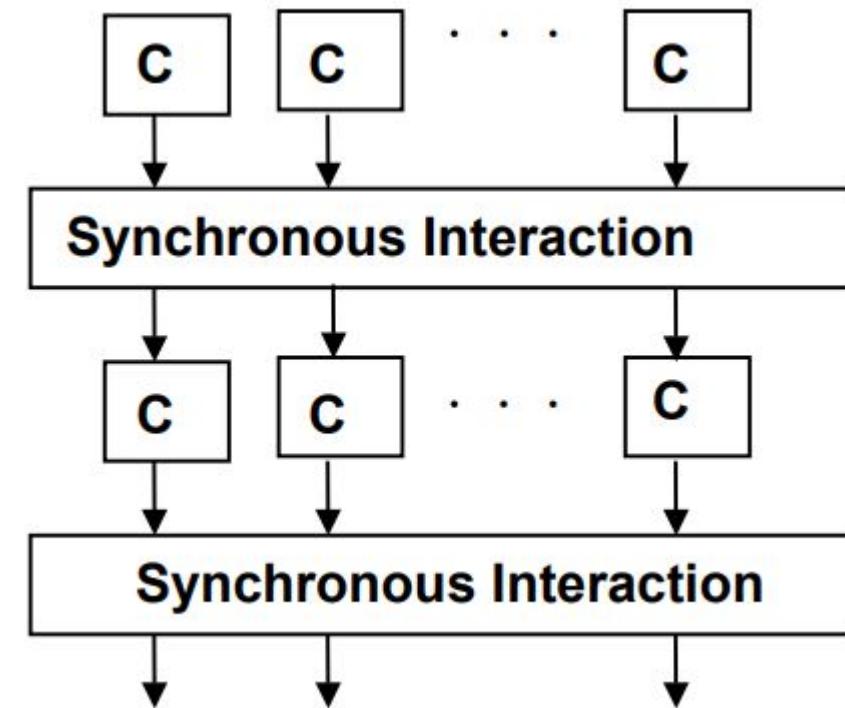
- Phase parallel
- Divide and conquer
- Pipeline
- Process farm
- Work pool

Note:

- The parallel program consists of number of super steps, and each super step has two phases : computation phase and interaction phase

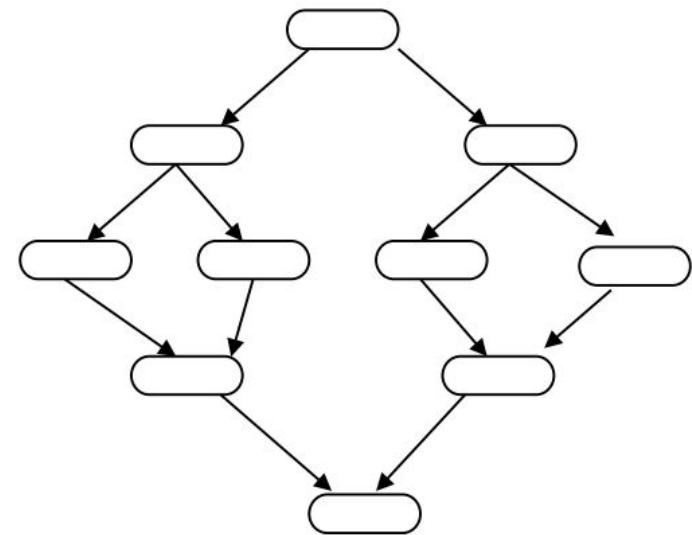
Phase Parallel Model

- The phase-parallel model offers a paradigm that is widely used in parallel programming.
- The parallel program consists of a number of supersteps, and each has two phases.
 - In a computation phase, multiple processes each perform an independent computation C.
 - In the subsequent interaction phase, the processes perform one or more synchronous interaction operations, such as a barrier or a blocking communication.
 - Then next superstep is executed.



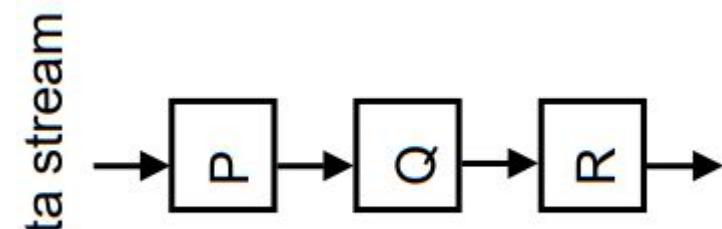
Divide and Conquer & Pipeline model

- A parent process divides its workload into several smaller pieces and assigns them to a number of child processes.
- The child processes then compute their workload in parallel and the results are merged by the parent.
- The dividing and the merging procedures are done recursively.
- This paradigm is very natural for computations such as quick sort.



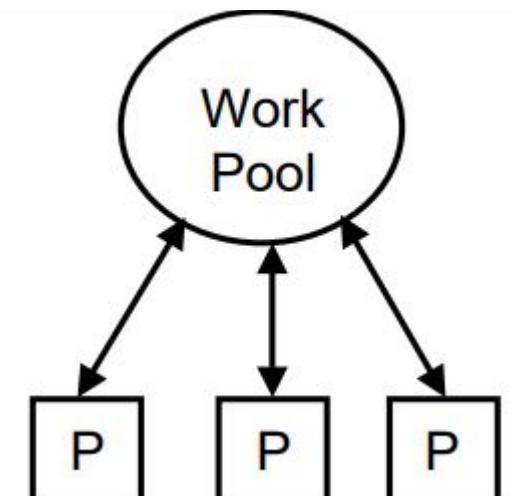
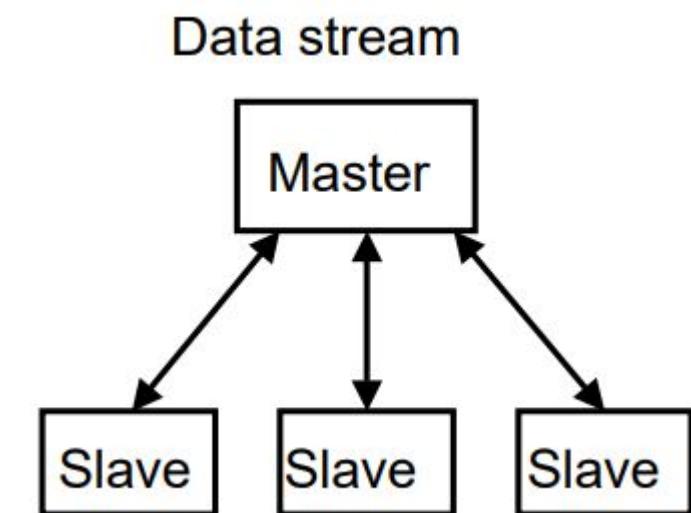
Pipeline

- In pipeline paradigm, a number of processes form a virtual pipeline.
- A continuous data stream is fed into the pipeline, and the processes execute at different pipeline stages simultaneously in an overlapped fashion.



Process Farm & Work Pool Model

- This paradigm is also known as the master-slave paradigm.
- A master process executes the essentially sequential part of the parallel program and spawns a number of slave processes to execute the parallel workload.
- When a slave finishes its workload, it informs the master which assigns a new workload to the slave.
- This is a very simple paradigm, where the coordination is done by the master.
- This paradigm is often used in a shared variable model.
- A pool of works is realized in a global data structure.
- A number of processes are created. Initially, there may be just one piece of work in the pool.
- Any free process fetches a piece of work from the pool and executes it, producing zero, one, or more new work pieces put into the pool. The parallel program ends when the work pool becomes empty.
- This paradigm facilitates load balancing, as the workload is dynamically allocated to free processes.



Parallel Program using Python

- A thread is basically an independent flow of execution. A single process can consist of multiple threads. Each thread in a program performs a particular task. For Example, when you are playing a game say FIFA on your PC, the game as a whole is a single process, but it consists of several threads responsible for playing the music, taking input from the user, running the opponent synchronously, etc.
- Threading is that it allows a user to run different parts of the program in a concurrent manner and make the design of the program simpler.
- Multithreading in Python can be achieved by importing the threading module.

Example:

```
import threading  
from threading import *
```

Parallel program using Threads in Python

```
# simplest way to use a Thread is to instantiate it with a target  
function and call start() to let it begin working.  
  
from threading import Thread,current_thread  
  
print(current_thread().getName())  
  
def mt():  
    print("Child Thread")  
  
    for i in range(11,20):  
        print(i*2)  
  
def disp():  
    for i in range(10):  
        print(i*2)  
  
child=Thread(target=mt)  
  
child.start()  
  
disp()  
  
print("Executing thread name : ",current_thread().getName())
```

```
from threading import Thread,current_thread  
  
class mythread(Thread):  
  
    def run(self):  
        for x in range(7):  
            print("Hi from child")  
  
        a = mythread()  
        a.start()  
        a.join()  
  
        print("Bye from",current_thread().getName())
```

Parallel program using Process in Python

```
import multiprocessing

def worker(num):
    print('Worker:', num)
    for i in range(num):
        print(i)
    return

jobs = []
for i in range(1,5):
    p = multiprocessing.Process(target=worker, args=(i+10,))
    jobs.append(p)
    p.start()
```

Concurrent Programming Paradigm

- Computing systems model the world, and the world contains actors that execute independently of, but communicate with, each other. In modelling the world, many (possibly) parallel executions have to be composed and coordinated, and that's where the study of concurrency comes in.
- There are two common models for concurrent programming: shared memory and message passing.
 - **Shared memory.** In the shared memory model of concurrency, concurrent modules interact by reading and writing shared objects in memory.
 - **Message passing.** In the message-passing model, concurrent modules interact by sending messages to each other through a communication channel. Modules send off messages, and incoming messages to each module are queued up for handling

Issues Concurrent Programming Paradigm

Concurrent programming is programming with multiple tasks. The major issues of concurrent programming are:

- Sharing computational resources between the tasks;
- Interaction of the tasks.

Objects shared by multiple tasks have to be safe for concurrent access. Such objects are called protected. Tasks accessing such an object interact with each other indirectly through the object.

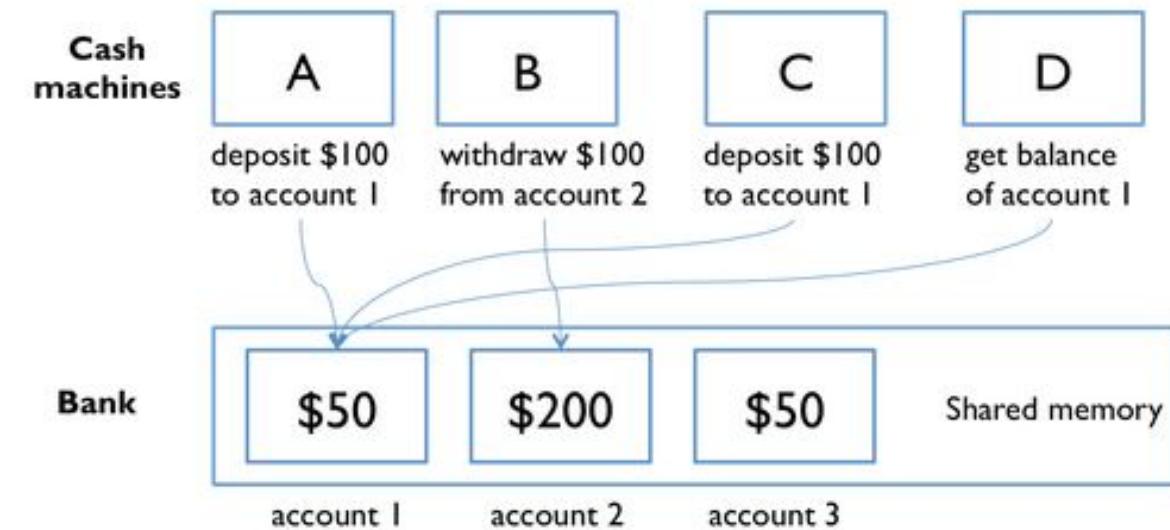
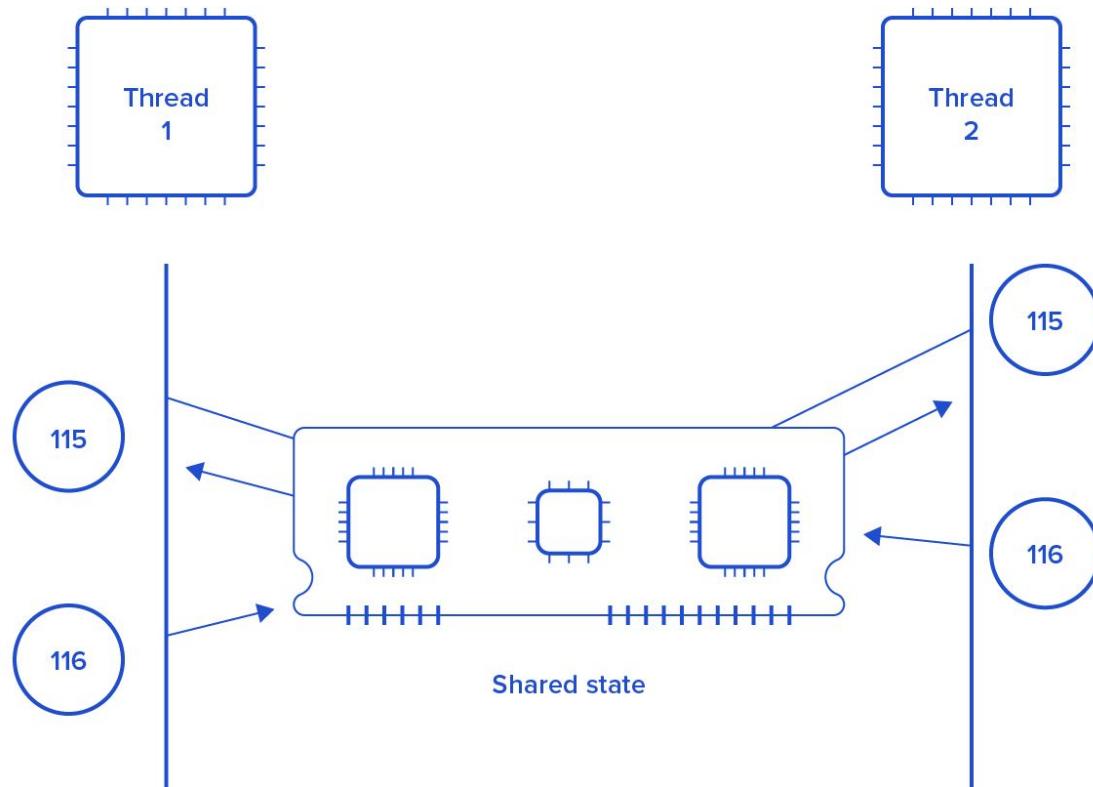
An access to the protected object can be:

- Lock-free, when the task accessing the object is not blocked for a considerable time;
- Blocking, otherwise.

Blocking objects can be used for task synchronization. To the examples of such objects belong:

- Events;
- Mutexes and semaphores;
- Waitable timers;
- Queues

Issues Concurrent Programming Paradigm



Race Condition

```
import threading
x = 0 # A shared value
COUNT = 100

def incr():
    global x
    for i in range(COUNT):
        x += 1
        print(x)

def decr():
    global x
    for i in range(COUNT):
        x -= 1
        print(x)

t1 = threading.Thread(target=incr)
t2 = threading.Thread(target=decr)
t1.start()
t2.start()
t1.join()
t2.join()
print(x)
```

Synchronization in Python

Locks:

Locks are perhaps the simplest synchronization primitives in Python. A Lock has only two states — locked and unlocked (surprise). It is created in the unlocked state and has two principal methods — acquire() and release(). The acquire() method locks the Lock and blocks execution until the release() method in some other co-routine sets it to unlocked.

R-Locks:

R-Lock class is a version of simple locking that only blocks if the lock is held by another thread. While simple locks will block if the same thread attempts to acquire the same lock twice, a re-entrant lock only blocks if another thread currently holds the lock.

Semaphore:

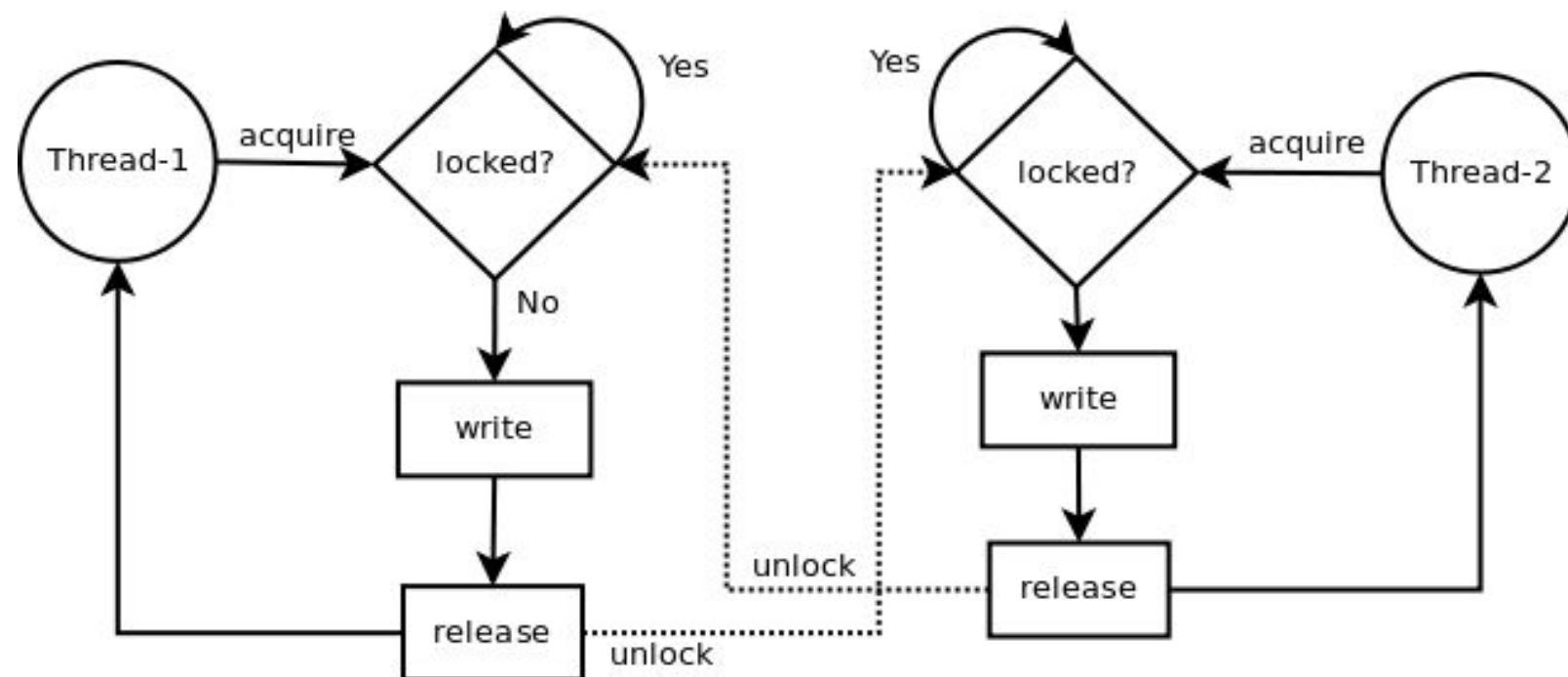
A semaphore has an internal counter rather than a lock flag, and it only blocks if more than a given number of threads have attempted to hold the semaphore. Depending on how the semaphore is initialized, this allows multiple threads to access the same code section simultaneously.

LOCK in python

Synchronization using LOCK

Locks have 2 states: locked and unlocked. 2 methods are used to manipulate them: acquire() and release(). Those are the rules:

1. if the state is unlocked: a call to acquire() changes the state to locked.
2. if the state is locked: a call to acquire() blocks until another thread calls release().
3. if the state is unlocked: a call to release() raises a RuntimeError exception.
4. if the state is locked: a call to release() changes the state to unlocked().



Synchronization in Python using Lock

```
import threading

x = 0    # A shared value
COUNT = 100
lock = threading.Lock()

def incr():
    global x
    lock.acquire()
    print("thread locked for increment cur x=",x)
    for i in range(COUNT):
        x += 1
        print(x)
    lock.release()
    print("thread release from increment cur x=",x)

def decr():
    global x
    lock.acquire()
    print("thread locked for decrement cur x=",x)
    for i in range(COUNT):
        x -= 1
        print(x)
    lock.release()
    print("thread release from decrement cur x=",x)

t1 = threading.Thread(target=incr)
t2 = threading.Thread(target=decr)
t1.start()
t2.start()
t1.join()
t2.join()
```

Synchronization in Python using RLock

```
import threading

class Foo(object):
    lock = threading.RLock()
    def __init__(self):
        self.x = 0
    def add(self,n):
        with Foo.lock:
            self.x += n
    def incr(self):
        with Foo.lock:
            self.add(1)
    def decr(self):
        with Foo.lock:
            self.add(-1)

def adder(f,count):
    while count > 0:
        f.incr()
        count -= 1

def subber(f,count):
    while count > 0:
        f.decr()
        count -= 1

# Create some threads and make sure it works
COUNT = 10
f = Foo()
t1 = threading.Thread(target=adder,args=(f,COUNT))
t2 = threading.Thread(target=subber,args=(f,COUNT))
t1.start()
t2.start()
t1.join()
t2.join()
print(f.x)
```

Synchronization in Python using Semaphore

```
import threading
import time
done = threading.Semaphore(0)
item = None
def producer():
    global item
    print "I'm the producer and I produce data."
    print "Producer is going to sleep."
    time.sleep(10)
    item = "Hello"
    print "Producer is alive. Signaling the consumer."
    done.release()
def consumer():
    print "I'm a consumer and I wait for data."
    print "Consumer is waiting."
    done.acquire()
    print "Consumer got", item
```

Synchronization in Python using event

```
import threading # A producer thread
import time
item = None
# A semaphore to indicate that an item is available
available = threading.Semaphore(0)
# An event to indicate that processing is complete
completed = threading.Event()
# A worker thread
def worker():
    while True:
        available.acquire()
        print "worker: processing", item
        time.sleep(5)
        print "worker: done"
        completed.set()
def producer():
    global item
    for x in range(5):
        completed.clear() # Clear the event
        item = x # Set the item
        print "producer: produced an item"
        available.release() # Signal on the semaphore
        completed.wait()
        print "producer: item was processed"
t1 = threading.Thread(target=producer)
t1.start()
t2 = threading.Thread(target=worker)
t2.setDaemon(True)
t2.start()
```

Producer and Consumer problem using thread

```
import threading,time,Queue  
  
items = Queue.Queue()  
  
# A producer thread  
  
def producer():  
    print "I'm the producer"  
  
    for i in range(30):  
        items.put(i)  
  
        time.sleep(1)  
  
# A consumer thread  
  
def consumer():  
    print "I'm a consumer", threading.currentThread().name  
  
    while True:  
        x = items.get()  
  
        print threading.currentThread().name,"got", x  
  
        time.sleep(5)  
  
        # Launch a bunch of consumers  
        cons = [threading.Thread(target=consumer)  
                for i in range(10)]  
  
        for c in cons:  
            c.setDaemon(True)  
            c.start()  
  
        # Run the producer  
        producer()
```

Producer and Consumer problem using thread

```
import threading
import time

# A list of items that are being produced. Note: it is actually
# more efficient to use a collections.deque() object for this.

items = []
# A condition variable for items
items_cv = threading.Condition()

def producer():
    print "I'm the producer"
    for i in range(30):
        with items_cv:      # Always must acquire the lock first
            items.append(i) # Add an item to the list
            items_cv.notify() # Send a notification signal
        time.sleep(1)

def consumer():
    print "I'm a consumer", threading.currentThread().name
    while True:
        with items_cv:      # Must always acquire the lock
            while not items: # Check if there are any items
                items_cv.wait() # If not, we have to sleep
            x = items.pop(0) # Pop an item off
            print threading.currentThread().name,"got", x
            time.sleep(5)
    cons = [threading.Thread(target=consumer)
            for i in range(10)]
    for c in cons:
        c.setDaemon(True)
        c.start()
    producer()
```

Network Programming Paradigm

Introduction

The Network paradigm involves thinking of computing in terms of a client, who is essentially in need of some type of information, and a server, who has lots of information and is just waiting to hand it out. Typically, a client will connect to a server and query for certain information. The server will go off and find the information and then return it to the client.

In the context of the Internet, clients are typically run on desktop or laptop computers attached to the Internet looking for information, whereas servers are typically run on larger computers with certain types of information available for the clients to retrieve. The Web itself is made up of a bunch of computers that act as Web servers; they have vast amounts of HTML pages and related data available for people to retrieve and browse. Web clients are used by those of us who connect to the Web servers and browse through the Web pages.

Network programming uses a particular type of network communication known as sockets. A socket is a software abstraction for an input or output medium of communication.

What is Socket?

- A socket is a software abstraction for an input or output medium of communication.
- Sockets allow communication between processes that lie on the same machine, or on different machines working in diverse environment and even across different continents.
- A socket is the most vital and fundamental entity. Sockets are the end-point of a two-way communication link.
- An endpoint is a combination of IP address and the port number.

For Client-Server communication,

- Sockets are to be configured at the two ends to initiate a connection,
- Listen for incoming messages
- Send the responses at both ends
- Establishing a bidirectional communication.

Socket Types

Datagram Socket

- A datagram is an independent, self-contained piece of information sent over a network whose arrival, arrival time, and content are not guaranteed. A datagram socket uses User Datagram Protocol (UDP) to facilitate the sending of datagrams (self-contained pieces of information) in an unreliable manner. Unreliable means that information sent via datagrams isn't guaranteed to make it to its destination.

Stream Socket:

- A stream socket, or connected socket, is a socket through which data can be transmitted continuously. A stream socket is more akin to a live network, in which the communication link is continuously active. A stream socket is a "connected" socket through which data is transferred continuously.

Socket in Python

```
sock_obj = socket.socket( socket_family, socket_type, protocol=0)
```

socket_family: - Defines family of protocols used as transport mechanism.

Either AF_UNIX, or

AF_INET (IP version 4 or IPv4).

socket_type: Defines the types of communication between the two end-points.

SOCK_STREAM (for connection-oriented protocols, e.g., TCP), or

SOCK_DGRAM (for connectionless protocols e.g. UDP).

protocol: We typically leave this field or set this field to zero.

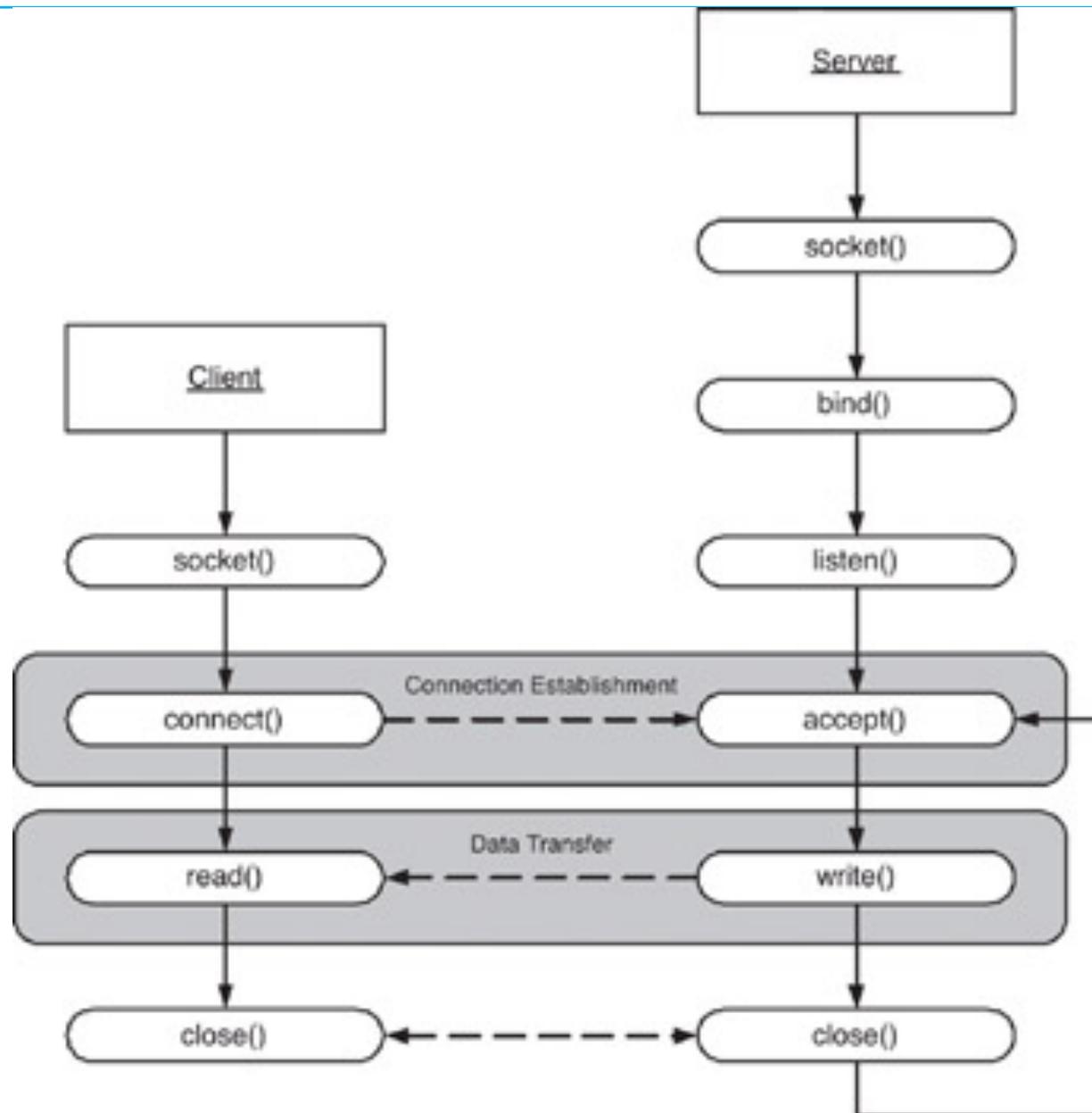
Example:

```
#Socket client example in python  
import socket  
  
#create an AF_INET, STREAM socket (TCP)  
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
print 'Socket Created'
```

Socket Creation

```
import socket  
  
import sys  
  
try:  
  
    #create an AF_INET, STREAM socket (TCP)  
  
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
  
except socket.error, msg:  
  
    print 'Failed to create socket. Error code: ' + str(msg[0]) + ' , Error message : ' + msg[1]  
  
    sys.exit();  
  
print 'Socket Created'
```

Client/server symmetry in Sockets applications



Socket in Python

To create a socket, we must use `socket.socket()` function available in the Python socket module, which has the general syntax as follows:

S = socket.socket(socket_family, socket_type, protocol=0)

`socket_family`: This is either AF_UNIX or AF_INET. We are only going to talk about INET sockets in this tutorial, as they account for at least 99% of the sockets in use.

`socket_type`: This is either SOCK_STREAM or SOCK_DGRAM.

Protocol: This is usually left out, defaulting to 0.

Client Socket Methods

Following are some client socket methods:

`connect()` : To connect to a remote socket at an address. An address format(`host, port`) pair is used for AF_INET address family.

Socket in Python

Server Socket Methods

`bind()`: This method binds the socket to an address. The format of address depends on socket family mentioned above(`AF_INET`).

`listen(backlog)` : This method listens for the connection made to the socket. The backlog is the maximum number of queued connections that must be listened before rejecting the connection.

`accept()` : This method is used to accept a connection. The socket must be bound to an address and listening for connections. The return value is a pair(`conn, address`) where `conn` is a new socket object which can be used to send and receive data on that connection, and `address` is the address bound to the socket on the other end of the connection.

General Socket in Python

`sock_object.recv():`

Use this method to receive messages at endpoints when the value of the protocol parameter is TCP.

`sock_object.send():`

Apply this method to send messages from endpoints in case the protocol is TCP.

`sock_object.recvfrom():`

Call this method to receive messages at endpoints if the protocol used is UDP.

`sock_object.sendto():`

Invoke this method to send messages from endpoints if the protocol parameter is UDP.

`sock_object.gethostname():`

This method returns hostname.

`sock_object.close():`

This method is used to close the socket. The remote endpoint will not receive data from this side.

Simple TCP Server

```
#!/usr/bin/python

#This is tcp_server.py script

import socket #line 1: Import socket module

s = socket.socket() #line 2: create a socket object
host = socket.gethostname() #line 3: Get current machine name
port = 9999 #line 4: Get port number for connection

s.bind((host,port)) #line 5: bind with the address

print "Waiting for connection..." #line 6: listen for connections

while True:
    conn,addr = s.accept() #line 7: connect and accept from client
    print 'Got Connection from', addr
    conn.send('Server Saying Hi')
    conn.close() #line 8: Close the connection
```

Simple TCP Client

```
#!/usr/bin/python

#This is tcp_client.py script

import socket

s = socket.socket()
host = socket.gethostname()          # Get current machine name
port = 9999                          # Client wants to connect to server's
                                      # port number 9999
s.connect((host,port))
print s.recv(1024)                  # 1024 is bufsize or max amount
                                      # of data to be received at once
s.close()
```

Simple UDP Server

```
#!/usr/bin/python

import socket

sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)          # For UDP

udp_host = socket.gethostname()                                # Host IP
udp_port = 12345                                              # specified port to connect

#print type(sock) =====> 'type' can be used to see type
# of any variable ('sock' here)

sock.bind((udp_host,udp_port))

while True:
    print "Waiting for client..."
    data,addr = sock.recvfrom(1024)                            #receive data from client
    print "Received Messages:",data," from",addr
```

Simple UDP Client

```
#!/usr/bin/python

import socket

sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)      # For UDP

udp_host = socket.gethostname()      # Host IP
udp_port = 12345                      # specified port to connect

msg = "Hello Python!"
print "UDP target IP:", udp_host
print "UDP target Port:", udp_port

sock.sendto(msg,(udp_host,udp_port))      # Sending message to UDP server
```

Logic + Control

What + How

**Clauses + resolution
database+interpreter**

facts+rules

If

**If X is a bird/an aeroplane, then X has wings.
Tweety is a bird**

Logic Programming Paradigm

- **Sub Topics**

- Definition - S1-SLO1
- First-class function, Higher-order function, Pure functions, Recursion- S1-SLO2
- *Packages: Kanren, SymPy – S2-SLO 1*
- *PySWIP, PyDatalog – S2-SLO 2*
- *Other languages: Prolog, ROOP, Janus S3-SLO 1*

Introduction

- It can be an abstract model of computation.
- Solve logical problems like puzzles, series
- Have **knowledge base** which we know before and along with the question
- you specify knowledge and how that knowledge is to be applied through a series of rules
- in logic programming, the common approach is to apply the methods of **resolution** and **unification**
- Knowledge is given to machine to produce result. Exp : **AL and ML code**

Introductions

- The *Logical Paradigm* takes a **declarative approach** to problem-solving.
- Various **logical assertions** about a situation are made, establishing all known facts.
- Then queries are made.
- The role of the computer becomes maintaining data and logical deduction.
- Programs are written in the language of some logic..
- Execution of a logic program is a theorem proving process; that is, computation is done by **logic inferences**
- Prolog (PROgramming in LOGic) is a) is a representative logic language
- **Exp :** Prolog, ROOP, Janus

What is a logic ?

- A logic is a language.
- It has syntax and semantics.
- More than a language, it has inference rules.
- **Syntax:**
 - The rules about how to form formulas;
 - This is usually the easy part of a logic.
- **Semantics:**
 - about the meaning carried by the formulas,
 - mainly in terms of logical consequences.
- **Inference rules**
 - Inference rules describe correct ways to derive

Features of Logical paradigms

- Computing takes place over the domain of all terms defined over a “universal” alphabet.
- Values are assigned to variables by means of automatically generated substitutions, called most general unifiers.
- These values may contain variables, called logical variables.
- The control is provided by a single mechanism: automatic backtracking.
- **Strength** - simplicity and Conciseness
- **Weakness** - has to do with the restrictions to one control mechanism and the use of a single data type.

History of Logic Programming (LP)

- Logic Programming has roots going back to early AI researchers like John McCarthy in the 50s & 60s
- Alain Colmerauer (France) designed Prolog as the first LP language in the early 1970s
- Bob Kowalski and colleagues in the UK evolved the language to its current form in the late 70s
- It's been widely used for many AI systems, but also for systems that need a fast, efficient and clean rule based engine
- The prolog model has also influenced the database community –
 - Exp datalog

General Overview

- Programs written in logic languages consist of declarations that are actually statements, or propositions, in symbolic logic.
- One of the essential characteristics of logic programming languages is their semantic
- The basic concept of this semantics is that there is a simple way to determine the meaning of each statement, and it does not depend on how the statement might be used to solve a problem.

Concepts of logic paradigm and theoretical background

- Includes both theoretical and fully implemented languages
- They all share the idea of interpreting computation as logical deduction.
- Notation Algorithm = Logic + Control
- Logic Programming uses facts and rules for solving the problem.
- That is why they are called the building blocks of Logic Programming.
- **Facts**
 - Actually, every logic program needs facts to work with so that it can achieve the given goal.
 - Facts basically are true statements about the program and data.
 - Exp : Delhi is the capital of India.
- **Rules**
 - are the constraints which allow us to make conclusions about the problem domain.
 - Basically written as logical clauses to express various facts.
 - Exp if we are building any game then all the rules must be defined.

Parts of Logical programming

1. A series of definitions/declarations that define the problem domain
2. Statements of relevant facts
3. Statement of goals in the form of a query

Advantages

- The system solves the problem, so the programming steps themselves are kept to a minimum;
- Proving the validity of a given program is simple.

Perspectives on logic programming

- Computations as Deduction
- Theorem Proving
- Non-procedural Programming
- Algorithms minus Control
- A Very High Level Programming Language
- A Procedural Interpretation of Declarative Specifications

Computation as Deduction

- Logic programming offers a slightly different paradigm for computation: computation is logical deduction
- It uses the language of logic to express data and programs.
- $\forall X, Y : X \text{ is the father of } Y \text{ if } X \text{ is a parent of } Y \text{ and } X \text{ is male}$
- Current logic programming languages use first order logic (FOL) which is often referred to as first order predicate calculus (FOPC).
- The first order refers to the constraint that we can quantify (i.e. generalize) over objects, but not over functions or relations.
- We can express "All elephants are mammals" but not
- "for every continuous function f , if $n < m$ and $f(n) < 0$ and $f(m) > 0$ then there exists an x such that $n < x < m$ and $f(x) = 0$ "

Theorem Proving

- Logic Programming uses the notion of an automatic theorem prover as an interpreter.
- The theorem prover derives a desired solution from an initial set of axioms.
- The proof must be a "constructive" one so that more than a true/false answer can be obtained
- E.G. The answer to
 - exists x such that $x = \sqrt{16}$
 - should be $x = 4$ or $x = -4$
 - rather than true

Non-procedural Programming

- Logic Programming languages are non-procedural programming languages
- A non-procedural language one in which one specifies what needs to be computed but not how it is to be done
- That is, one specifies:
 - the set of objects involved in the computation
 - the relationships which hold between them
 - the constraints which must hold for the problem to be solved
 - and leaves it up the language interpreter or compiler to decide how to satisfy the constraints

A Declarative Example

- Here's a simple way to specify what has to be true if X is the smallest number in a list of numbers L
 1. X has to be a member of the list L
 2. There can be list member X2 such that $X2 < X$

We need to say how we determine that some X is a member of a list

1. No X is a member of the empty list
2. X is a member of list L if it is equal to L's head
3. X is a member of list L if it is a member of L's tail.

Use logic to do reasoning

- Example: Given information about fatherhood and motherhood, determine grand parent relationship
- E.g. Given the information called facts
 - John is father of Lily
 - Kathy is mother of Lily
 - Lily is mother of Bill
 - Ken is father of Karen
- Who are grand parents of Bill?
- Who are grand parents of Karen?

Example

domains

being = symbol

predicates

animal(being) % all animals are beings

dog(being) % all dogs are beings

die(being) % all beings die

clauses

animal(X) :- dog(X) % all dogs are animals

dog(fido). % fido is a dog

die(X) :- animal(X) % all animals die

Logic Operators

Name	Symbol	Example	Meaning
negation	\neg	$\neg a$	not a
conjunction	\cap	$a \cap b$	a and b
disjunction	\cup	$a \cup b$	a or b
equivalence	\equiv	$a \equiv b$	a is equivalent to b
implication	\supset \subset	$a \supset b$ $a \subset b$	a implies b b implies a
universal	$\forall X.P$		For all X, P is true
existential	$\exists X.P$		There exists a value of X such that P is true

- Equivalence means that both expressions have identical truth tables
- Implication is like an if-then statement
 - if a is true then b is true
 - note that this does not necessarily mean that if a is false that b must also be false
- Universal quantifier says that this is true no matter what x is
- Existential quantifier says that there is an X that fulfills the statement

$\forall X.(\text{woman}(X) \supset \text{human}(X))$
 – if X is a woman, then X is a human

$\exists X.(\text{mother}(\text{mary}, X) \cap \text{male}(X))$
 – Mary has a son (X)

Example Statements

Consider the following knowledge:

Bob is Fred's father \square father(Bob, Fred)

Sue is Fred's mother \square mother(Sue, Fred)

Barbara is Fred's sister \square sister(Barbara, Fred)

Jerry is Bob's father \square father(Jerry, Bob)

And the following rules:

A person's father's father is the person's grandfather

A person's father or mother is that person's parent

A person's sister or brother is that person's sibling

If a person has a parent and a sibling, then the sibling has the same parent

These might be captured in first-order predicate calculus as:

$$\forall x, y, z : \text{if } \text{father}(x, y) \text{ and } \text{father}(y, z) \text{ then } \text{grandfather}(x, z)$$
$$\forall x, y : \text{if } \text{father}(x, y) \text{ or } \text{mother}(x, y) \text{ then } \text{parent}(x, y)$$
$$\forall x, y : \text{if } \text{sister}(x, y) \text{ or } \text{brother}(x, y) \text{ then } \text{ sibling}(x, y) \text{ and } \text{ sibling}(y, x)$$
$$\forall x, y, z : \text{if } \text{parent}(x, y) \text{ and } \text{ sibling}(y, z) \text{ then } \text{parent}(x, z)$$

We would rewrite these as

$$\text{grandfather}(x, z) \subseteq \text{father}(x, y) \text{ and } \text{father}(y, z)$$
$$\text{parent}(x, y) \subseteq \text{father}(x, y)$$
$$\text{parent}(x, y) \subseteq \text{mother}(x, y) \quad \text{etc}$$

Kanren

- It provides us a way to simplify the way we made code for business logic.
- It lets us express the logic in terms of rules and facts.
- The following command will help you install kanren –
 - `pip install kanren`

Matching mathematical expressions

```
from kanren import run, var, fact  
from kanren.assoccomm import eq_assoccomm as eq  
from kanren.assoccomm import commutative, associative  
add = 'add'  
mul = 'mul'  
fact(commutative, mul)  
fact(commutative, add)  
fact(associative, mul)  
fact(associative, add)  
a, b = var('a'), var('b')  
Original_pattern = (mul, (add, 5, a), b)  
exp1 = (mul, 2, (add, 3, 1))  
exp2 = (add, 5, (mul, 8, 1))  
print(run(0, (a,b), eq(original_pattern, exp1)))  
print(run(0, (a,b), eq(original_pattern, exp2)))
```

- **Output**

- ((3,2))
- ()

- **Reference :**

- [https://www.tutorialspoint.com/artificial intelligence with python/artificial intelligence with python logic programming.htm](https://www.tutorialspoint.com/artificial_intelligence_with_python/artificial_intelligence_with_python_logic_programming.htm)

SymPy Overview

Simplification

simplify

Polynomial/Rational Function Simplification

Trigonometric Simplification

Powers

Exponentials and logarithms

Special Functions

Example: Continued Fractions

Calculus

Derivatives

Integrals

Limits

Series Expansion

Finite differences

Solvers

A Note about Equations

Solving Equations Algebraically

Solving Differential Equations

Matrices

Basic Operations

Basic Methods

Matrix Constructors

Advanced Methods

Possible Issues

Reference:

- <https://www.geeksforgeeks.org/python-getting-started-with-sympy-module/>
- <https://docs.sympy.org/latest/tutorial/index.html>

SymPy module

- **Sympy** is a Python library for symbolic mathematics.
- It aims to become a full-featured computer algebra system (CAS)
- While keeping the code as simple as possible in order to be comprehensible and easily extensible.
- SymPy is written entirely in Python.
- **Installing sympy module:**
 - **Pip install sympy**

Example

```
from sympy import * x = Symbol('x')
y = Symbol('y')
z = (x + y) + (x-y)
print("value of z is :" + str(z))
```

Output:

value of z is : $2*x$

Find derivative, integration, limits, quadratic equation

```
# make a symbol
x = Symbol('x')
# take the derivative of sin(x)*e ^ x
ans1 = diff(sin(x)*exp(x), x)
print("derivative of sin(x)*e ^ x : ", ans1)
# Compute (e ^ x * sin(x)+ e ^ x * cos(x))dx
ans2 = integrate(exp(x)*sin(x) + exp(x)*cos(x), x)
print("indefinite integration is : ", ans2)
# Compute definite integral of sin(x ^ 2)dx
# in b / w interval of ? and ?? .
ans3 = integrate(sin(x**2), (x, -oo, oo))
print("definite integration is : ", ans3)
# Find the limit of sin(x) / x given x tends to 0
ans4 = limit(sin(x)/x, x, 0)
print("limit is : ", ans4)
# Solve quadratic equation like, example : x ^ 2?2 = 0
ans5 = solve(x**2 - 2, x)
print("roots are : ", ans5)
```

Output :

```
derivative of sin(x)*e^x : exp(x)*sin(x) + exp(x)*cos(x)
indefinite integration is : exp(x)*sin(x)
definite integration is : sqrt(2)*sqrt(pi)/2
limit is : 1
roots are : [-sqrt(2), sqrt(2)]
```

[Sympy Live Shell Demo](#)

<https://docs.sympy.org/latest/tutorial/intro.html#what-is-symbolic-computation>

Datalog Concepts

- pyDatalog is a powerful language with very few syntactic elements, mostly coming from Python :
 - Variables and expressions
 - Loops
 - Facts
 - Logic Functions and dictionaries
 - Aggregate functions
 - Literals and sets
 - Tree, graphs and recursive algorithms
 - 8-queen problem

Reference

- <https://sites.google.com/site/pydatalog/Online-datalog-tutorial>

PySwip Introduction

- PySwip is a Python - SWI-Prolog bridge enabling to query [SWI-Prolog](#) in your Python programs.
- It features an (incomplete) SWI-Prolog foreign language interface, a utility class that makes it easy querying with Prolog and also a Pythonic interface.
- Since PySwip uses SWI-Prolog as a shared library and ctypes to access it,
- it doesn't require compilation to be installed.
- Reference :
 - <https://pypi.org/project/pyswip/>

SRM Institute of Science and Technology
School of Computing

Advanced Programming Practice-18CSC207J

Paradigm types

Unit IV

- Functional Programming Paradigm [Text Book :1]
- Logic Programming Paradigm [Text Book : 1& 3]
- Dependent Type Programming Paradigm
- Network Programming Paradigm [Text Book :4]

Text Book:

1. Elad Shalom, A Review of Programming Paradigms throughout the History: With a suggestion Toward a Future Approach, Kindle Edition, 2018
2. John Goerzen, Brandon Rhodes, Foundations of Python Network Programming: The comprehensive guide to building network applications with Python, 2nd ed., Kindle Edition, 2010
3. Amit Saha, Doing Math with Python: Use Programming to Explore Algebra, Statistics,Calculus and More, Kindle Edition, 2015

Functional Programming Paradigm

Unit-III (15 Session)

Session 11-15 cover the following topics:-

- *Definition - S11-SLO1*
- *Sequence of Commands – S11-SLO2*
- *map(), reduce(), filter(), lambda – S12-SLO1*
- *partial, functools – S12-SLO2*
- *Other languages:F#, Clojure, Haskell – S13-SLO1*
- *Demo: Functional Programming in Python - S13-SLO2*

Lab 9: Functional Programming (Case Study) (S14-15)

Assignment : Comparative study of Functional programming in F#, Clojure, Haskell

- **TextBook:** Shalom, Elad. A Review of Programming Paradigms Throughout the History: With a Suggestion Toward a Future Approach, Kindle Edition

Functional Programming Paradigm (TF1)

Definition

- Mainly treat **computation to evaluate mathematical Functions**
- Avoids changing-state and mutable data
- Called as **declarative programming** paradigm
- Output depends on argument passing
- Calling same function several times with same values produces same result
- Uses expressions or declarations rather than statements as in imperative languages

Concepts

- It views all subprograms as **functions** in the mathematical sense
- Take in arguments and return a single solution.
- Solution returned is based entirely on input, and the time at which a function is called has no relevance.
- The computational model is therefore one of function application and reduction.

Characteristics of Functional Programming

- Functional programming method focuses on results, not the process
- Emphasis is on what is to be computed
- Data is immutable
- Functional programming Decompose the problem into 'functions'
- It is built on the concept of mathematical functions which uses conditional expressions and recursion to do perform the calculation
- It does not support iteration like loop statements and conditional statements like If-Else

History

- The foundation for Functional Programming is Lambda Calculus. It was developed in the 1930s for the functional application, definition, and recursion
- LISP was the first functional programming language. McCarthy designed it in 1960
- In the late 70's researchers at the University of Edinburgh defined the ML(Meta Language)
- In the early 80's Hope language adds algebraic data types for recursion and equational reasoning
- In the year 2004 Innovation of Functional language 'Scala.'

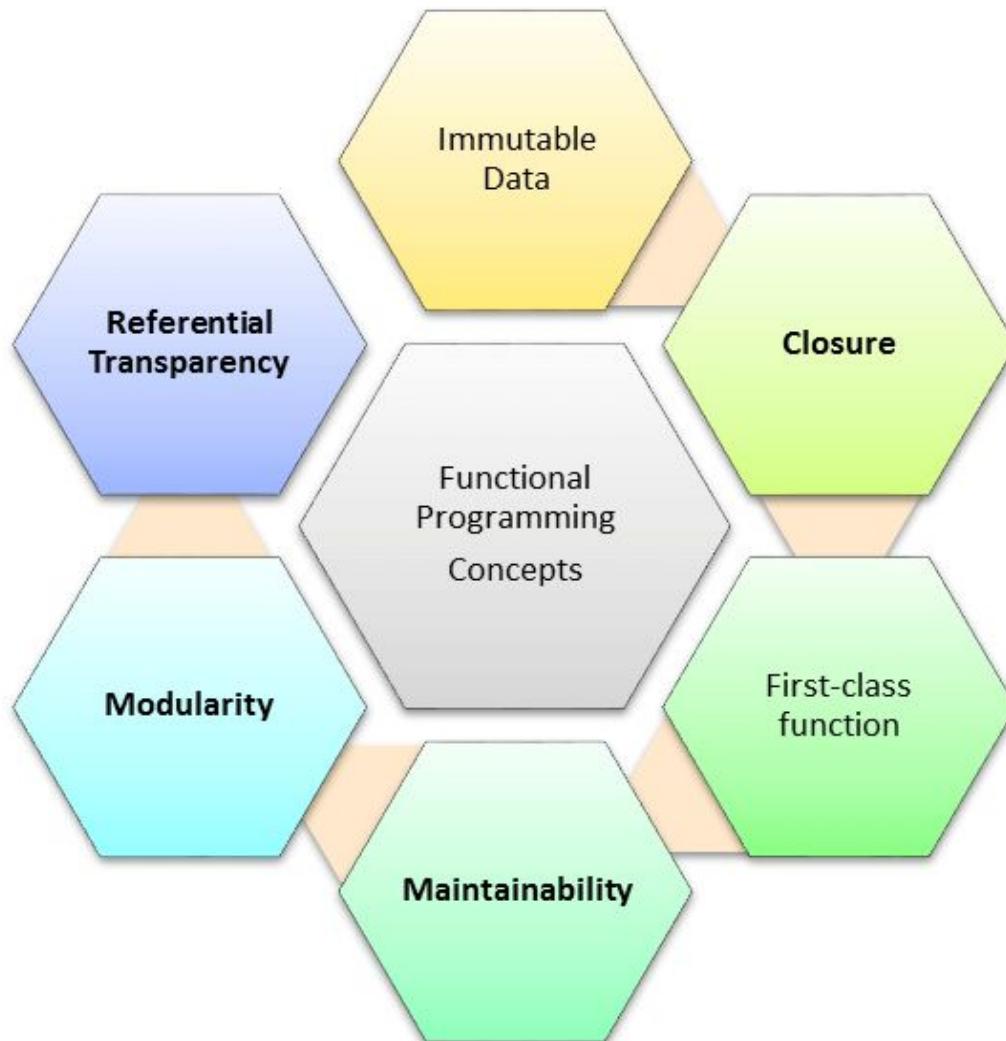
Real Time applications

- Database processing
- Financial modeling
- Statistical analysis and
- Bio-informatics

Functional Programming Languages

- Haskell
- SML
- Clojure
- Scala
- Erlang
- Clean
- F#
- ML/OCaml Lisp / Scheme
- XSLT
- SQL
- Mathematica

Basic Functional Programming Terminology and Concepts



Functional Vs Procedural

S.No	Functional Paradigms	Procedural Paradigm
1	Treats <u>computation</u> as the evaluation of <u>mathematical functions</u> avoiding <u>state</u> and <u>mutable</u> data	Derived from structured programming, based on the concept of <u>modular programming</u> or the <i>procedure call</i>
2	<u>Main traits are Lambda calculus, compositionality, formula, recursion, referential transparency</u>	Main traits are <u>Local variables</u> , sequence, selection, <u>iteration</u> , and <u>modularization</u>
3	Functional programming focuses on expressions	Procedural programming focuses on statements
4	Often recursive. Always returns the same output for a given input.	The output of a routine does not always have a direct correlation with the input.
5	Order of evaluation is usually undefined.	Everything is done in a specific order.
6	Must be stateless. i.e. No operation can have side effects.	Execution of a routine may have side effects.
7	Good fit for parallel execution, Tends to emphasize a divide and conquer approach.	Tends to emphasize implementing solutions in a linear fashion.

Functional Vs Object-oriented Programming

S.No	Functional Paradigms	Object Oriented Paradigm
1	FP uses Immutable data.	OOP uses Mutable data.
2	Follows Declarative Programming based Model.	Follows Imperative Programming Model.
3	What it focuses is on: "What you are doing. in the programme."	What it focuses is on "How you are doing your programming."
4	Supports Parallel Programming.	No supports for Parallel Programming.
5	Its functions have no-side effects.	Method can produce many side effects.
6	Flow Control is performed using function calls & function calls with recursion.	Flow control process is conducted using loops and conditional statements.
7	Execution order of statements is not very important.	Execution order of statements is important.
8	Supports both "Abstraction over Data" and "Abstraction over Behavior."	Supports only "Abstraction over Data".

Example

Functional Programming

```
num = 1  
def function_to_add_one(num):  
    num += 1  
    return num  
  
function_to_add_one(num)  
function_to_add_one(num)  
function_to_add_one(num)  
function_to_add_one(num)  
function_to_add_one(num)
```

#Final Output: 2

Procedural Programming

```
num = 1  
def procedure_to_add_one():  
    global num  
    num += 1  
    return num  
  
procedure_to_add_one()  
procedure_to_add_one()  
procedure_to_add_one()  
procedure_to_add_one()  
procedure_to_add_one()
```

#Final Output: 6

Features of Functional paradigms

- **First-class functions** – accept another function as an argument or return a function
- **Pure functions** - they are functions without side effects
- **Recursion** - allows writing smaller algorithms and operating by looking only at the inputs to a function
- **Immutable variables** - variables that cannot be changed
- **Non-strict evaluation** - allows having variables that have not yet been computed
- **Statements** - evaluable pieces of code that have a return value
- **Pattern matching** - allows better type-checking and extracting elements from an object

First-class function

- First-class functions are functions treated as objects themselves,
- Meaning they can be passed as a parameter to another function, returned as a result of a function
- It is said that a programming language has first-class functions if it treats functions as first-class citizens.

Properties of First-Class Functions

- It is an instance of an Object type
- Functions can be stored as variable
- Pass First Class Function as argument of some other functions
- Return Functions from other function
- Store Functions in lists, sets or some other data structures.

Functions as first-class objects in python

- Using functions as first-class objects means to use them in the same manner that you use data.
- So, you can pass them as parameters like passing a function to another function as an argument.
- `>>> list(map(int, ["1", "2", "3"])) [1, 2, 3]`

Higher-order function

- takes one or more functions as an input
- outputs a function
- Exp : Map Function
- Consider a function that prints a line multiple times:
 - Example Code

```
def write_repeat(message, n):  
    for i in range(n):  
        print(message)  
    write_repeat('Hello', 5)
```

Pure Functions

- **is idempotent** — returns the same result if provided the same arguments,
- has no side effects.
- If a function uses an object from a higher scope or random numbers, communicates with files and so on, it might be *impure*
- Since its result doesn't depend only on its arguments.

Example

```
def multiply_2_pure(numbers):
    new_numbers = []
    for n in numbers:
        new_numbers.append(n * 2)
    return new_numbers
```

```
original_numbers = [1, 3, 5, 10]
changed_numbers =
    multiply_2_pure(original_numbers)
print(original_numbers) # [1, 3, 5, 10]
print(changed_numbers) # [2, 6, 10, 20]
```

Anonymous Functions

- Anonymous (lambda) functions can be very convenient for functional programming constructs.
- They don't have names and usually are created ad-hoc, with a single purpose.
- Exp1: `lambda x, y: x + y`
- Exp 2: `>>> f = lambda x, y: x + y`
`>>> def g(x, y):`
`return x + y`

Examples

Anonymous function assigned to a variable. Easy to pass around and invoke when needed.

```
const myVar = function(){console.log('Anonymous function here!')}  
myVar()
```

Anonymous function as argument

- setInterval(function(){console.log(new Date().getTime())}, 1000);

Anonymous functions within a higher order function

```
function mealCall(meal){  
    return function(message){  
        return console.log(message + " " + meal + '!')  
    }  
}
```

```
const announceDinner = mealCall('dinner')
```

```
const announceLunch = mealCall('breakfast')
```

```
announceDinner('hey!, come and get your')
```

```
announceLunch('Rise and shine! time for')
```

Example using Python

- Anonymous functions are split into two types: lambda functions and closures.
- Functions are made of **four parts**: name, parameter list, body, and return

Example :

A = 5

```
def impure_sum(b):
    return b + A
def pure_sum(a, b):
    return a + b
print(impure_sum(6))
>> 11
print(pure_sum(4, 6))
>> 10
```

Example Code

Function for computing the average of two numbers:

```
(defun avg(X Y) (/ (+ X Y)  
2.0))
```

Function is called by:

```
> (avg 10.0 20.0)
```

Function returns:

```
15.0
```

Functional style of getting a sum of a list:

```
new_lst = [1, 2, 3, 4]  
def sum_list(lst):  
    if len(lst) == 1:  
        return lst[0]  
    else:  
        return lst[0] + sum_list(lst[1:])  
print(sum_list(new_lst))
```

or the pure functional way in python using higher order function

```
import functools  
print(functools.reduce(lambda x, y: x + y, new_lst))
```

Immutable variables

- Immutable variable (object) is a variable whose state cannot be modified once it is created.
- In contrast, a mutable variable can be modified after it is created
- **Exp**
- String str = “A Simple String.”;
- str.toLowerCase();

Other Examples

- int i = 12; //int is a primitive type
- i = 30; //this is ok
- final int j = 12;
- j = 30;
- final MyClass c = new MyClass();
- m.field = 100; //this is ok;
- m = new MyClass();

Functional programming tools

- **filter(*function, sequence*)**

```
def f(x): return x%2 != 0 and x%3 ==0  
filter(f, range(2,25))
```

- **map(*function, sequence*)**

- call function for each item
- return list of return values

- **reduce(*function, sequence*)**

- return a single value
- call binary function on the first two items
- then on the result and next item
- iterate

Lambda Expression

```
# Using `def` (old way).
```

```
def old_add(a, b):  
    return a + b
```

```
# Using `lambda` (new way).
```

```
new_add = lambda a, b: a + b  
old_add(10, 5) == new_add(10, 5)  
>> True
```

```
unsorted = [('b', 6), ('a', 10), ('d', 0), ('c', 4)]
```

```
print(sorted(unsorted, key=lambda x: x[1]))
```

```
>> [('d', 0), ('c', 4), ('b', 6), ('a', 10)]
```

Map Function

- Takes in an iterable (ie. list), and creates a new iterable object, a special map object.
- The new object has the first-class function applied to every element.

Pseudocode for map.

```
def map(func, seq):
    return Map(
        func(x)
        for x in seq
    )
```

Example:

```
values = [1, 2, 3, 4, 5]
```

```
add_10 = list(map(lambda x: x + 10, values))
add_20 = list(map(lambda x: x + 20, values))
```

```
print(add_10)
>> [11, 12, 13, 14, 15]
```

```
print(add_20)
>> [21, 22, 23, 24, 25]
```

Filter Function

- Takes in an iterable (ie. list), and creates a new iterable object
- The new object has the first-class function applied to every element.

Pseudocode for map.

```
return Map(  
    x for x in seq  
    if evaluate(x) is True  
)
```

Example:

```
even = list(filter(lambda x: x % 2 == 0,  
                  values))  
odd = list(filter(lambda x: x % 2 == 1, values))
```

```
print(even)  
>> [2, 4, 6, 8, 10]
```

```
print(odd)  
>> [1, 3, 5, 7, 9]
```

Advantages

- Allows you to avoid confusing problems and errors in the code
- Easier to test and execute Unit testing and debug FP Code.
- Parallel processing and concurrency
- Hot code deployment and fault tolerance
- Offers better modularity with a shorter code
- Increased productivity of the developer
- Supports Nested Functions
- Functional Constructs like Lazy Map & Lists, etc.
- Allows effective use of Lambda Calculus

Disadvantages

- Perhaps less efficiency
- Problems involving many variables, or a lot of sequential activity are sometimes easier to handle imperatively
- Functional programming paradigm is not easy, so it is difficult to understand for the beginner
- Hard to maintain as many objects evolve during the coding
- Needs lots of mocking and extensive environmental setup
- Re-use is very complicated and needs constantly refactoring
- Objects may not represent the problem correctly

Transforming code from imperative to functional

1. Introduce higher-order functions.
2. Convert existing methods into pure functions.
3. Convert loops over to recursive/tail-recursive methods (if possible).
4. Convert mutable variables into immutable variables.
5. Use pattern matching (if possible).

Mathematical Background

- For example, if $f(x)=x^2$ and x is 3, the ordered pair is $(3, 9)$.
- A function is defined by its set of inputs, called the domain.
- A set containing the set of outputs, and possibly additional elements as members, is called its codomain.
- The set of all input-output pairs is called its graph.

Mathematical Background

General Concepts

- **Notation**

$$F(x) = y$$

x , y -> Arguments or
parameters

x -> domain and
y->codomain

Types:

- Injective if $f(a) \neq f(b)$
- Surjective if $f(X) = Y$
- Bijective (support both)

Functional Rules

1. $(f+g)(x)=f(x)+g(x)$
2. $(f-g)(x)=f(x)-g(x)$
3. $(f*g)(x)=f(x)*g(x)$
4. $(f/g)(x)=f(x)/g(x)$
5. $(gof)(x)=g(f(x))$
6. $f f^{-1}=\text{id}_y$

Scenario1

- What if we wanted to write to a file 5 times, or log the message 5 times? Instead of writing 3 different functions that all loop, we can write 1 Higher Order Function that accepts those functions as an argument:

```
def hof_write_repeat(message, n, action):  
    for i in range(n):  
        action(message)
```

```
hof_write_repeat('Hello', 5, print)
```

```
# Import the logging library  
import logging  
  
# Log the output as an error instead  
hof_write_repeat('Hello', 5, logging.error)
```

Scenario2

- Imagine that we're tasked with creating functions that increment numbers in a list by 2, 5, and 10. So instead of creating many different increment functions, we create 1 Higher Order Function:

```
def hof_add(increment):  
    # Create a function that loops and adds  
    # the increment  
    def add_increment(numbers):  
        new_numbers = []  
        for n in numbers:  
            new_numbers.append(n +  
                increment)  
        return new_numbers  
    # We return the function as we do any  
    # other value  
    return add_increment  
add2=hof_add(2)  
print(add2([23, 88])) # [25, 90]  
add5 = hof_add(5)  
print(add5([23, 88])) # [28, 93]  
add10 = hof_add(10)  
print(add10([23, 88])) # [33, 98]
```

Sample Scenario

1) (The MyTriangle module) Create a module named MyTriangle that contains the following two functions:

Returns true if the sum of any two sides is greater than the third side.

def isValid(side1, side2, side3):

Returns the area of the triangle.

def area(side1, side2, side3):

Write a test program that reads three sides for a triangle and computes the area if the input is valid. Otherwise, it displays that the input is invalid.

2) A prime number is called a Mersenne prime if it can be written in the form for some positive integer p. Write a program that finds all Mersenne primes with and displays the output as follows:

p $2^p - 1$

2 3

3 7

5 31

Sample Scenarios

- 3) Write a function named `ack` that evaluates the Ackermann function. Use your function to evaluate `ack(3, 4)`, which should be 125.

The Ackermann function, $A(m, n)$, is defined:

$$A(m, n) = \begin{cases} n + 1 & \text{if } m = 0 \\ A(m - 1, 1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m - 1, A(m, n - 1)) & \text{if } m > 0 \text{ and } n > 0. \end{cases}$$

Thank you

Summary

Symbolic Programming Paradigm

Introduction

Symbolic computation deals with the computation of mathematical objects symbolically. This means that the mathematical objects are represented exactly, not approximately, and mathematical expressions with unevaluated variables are left in symbolic form.

It Covers the following:

- As A calculator and symbols
- Algebraic Manipulations - Expand and Simplify
- Calculus – Limits, Differentiation, Series , Integration
- Equation Solving – Matrices

Calculator and Symbols

Rational - $\frac{1}{2}$, or 5/2

```
>>import sympy as sym  
>>a = sym.Rational(1, 2)  
>>a
```

Answer will be $\frac{1}{2}$

Constants like pi,e

```
>>sym.pi**2  
>>sym.pi.evalf()  
>>(sym.pi + sym.exp(1)).evalf()
```

Answer is π^{**2}

Answer is 3.14159265358979

Answer is 5.85987448204884

X AND Y

```
>>x = sym.Symbol('x')  
>>y = sym.Symbol('y')  
>>x + y + x - y
```

Answer is $2*x$

Algebraic Manipulations

EXPAND (X+Y)3 = X+3X^2Y+3XY^2+Y**

>> sym.expand((x + y) ** 3) Answer is $x^{**3} + 3*x^{**2}*y + 3*x*y^{**2} + y^{**3}$

>> 3 * x * y ** 2 + 3 * y * x ** 2 + x ** 3 + y ** 3 Answer is $x^{**3} + 3*x^{**2}*y + 3*x*y^{**2} + y^{**3}$

WITH TRIGNOMETRY LIKE SIN,COSINE

eg. $\cos(X+Y) = -\sin(X)\sin(Y)+\cos(X)\cos(Y)$

>> sym.expand(sym.cos(x + y), trig=True) Answer is $-\sin(x)\sin(y) + \cos(x)\cos(y)$

SIMPLIFY

$$(X+X*Y/X)=Y+1$$

>> sym.simplify((x + x * y) / x) Answer is: $y+1$

Calculus

LIMITS compute the limit of

`limit(function, variable, point)`

`limit(sin(x)/x , x, 0) =1`

Differentiation

`diff(func,var) eg diff(sin(x),x)=cos(x)`

`diff(func,var,n) eg`

Series

`series(expr,var)`

`series(cos(x),x) = 1-x/2+x/24+o(x)`

Integration

`Integrate(expr,var)`

`Integrate(sin(x),x) = -cos(x)`

Example

Example:

```
In [23]: sym.expand(sym.cos(x + y), trig=True)
```

```
Out[23]: -sin(x)*sin(y) + cos(x)*cos(y)
```

```
In [24]: sym.limit(sym.sin(x) / x, x, 0)
```

```
Out[24]: 1
```

```
In [26]: sym.diff(sym.sin(x), x)
```

```
Out[26]: cos(x)
```

```
In [27]: sym.diff(sym.sin(2 * x), x)
```

```
Out[27]: 2*cos(2*x)
```

```
In [28]: sym.diff(sym.tan(x), x)
```

```
Out[28]: tan(x)**2 + 1
```

```
In [29]: sym.diff(sym.sin(2 * x), x, 1)
```

```
Out[29]: 2*cos(2*x)
```

```
In [30]: sym.diff(sym.sin(2 * x), x, 2)
```

```
Out[30]: -4*sin(2*x)
```

```
In [31]: sym.diff(sym.sin(2 * x), x, 3)
```

```
Out[31]: -8*cos(2*x)
```

```
In [31]: sym.diff(sym.sin(2 * x), x, 3)
```

```
Out[31]: -8*cos(2*x)
```

```
In [32]: sym.series(sym.cos(x), x)
```

```
Out[32]: 1 - x**2/2 + x**4/24 + O(x**6)
```

```
In [34]: sym.integrate(6 * x ** 5, x)
```

```
Out[34]: x**6
```

```
In [35]: sym.integrate(sym.sin(x), x)
```

```
Out[35]: -cos(x)
```

```
In [36]: sym.integrate(sym.log(x), x)
```

```
Out[36]: x*log(x) - x
```

```
In [37]: sym.integrate(2 * x + sym.sinh(x), x)
```

```
Out[37]: x**2 + cosh(x)
```

```
In [37]: sym.integrate(2 * x + sym.sinh(x), x)
```

```
Out[37]: x**2 + cosh(x)
```

```
In [38]: sym.integrate(sym.exp(-x ** 2) * sym.erf(x), x)
```

```
Out[38]: sqrt(pi)*erf(x)**2/4
```

```
In [39]: sym.integrate(x**3, (x, -1, 1))
```

```
Out[39]: 0
```

```
In [40]: sym.integrate(sym.sin(x), (x, 0, sym.pi / 2))
```

```
Out[40]: 1
```

Example

Example:

```
In [43]: sym.solveset(x ** 4 - 1, x)
```

```
Out[43]: {-1, 1, -I, I}
```

```
In [44]: sym.solveset(sym.exp(x) + 1, x)
```

```
Out[44]: ImageSet(Lambda(_n, I*(2*_n*pi + pi)), S.Integers)
```

```
In [46]: solution = sym.solve((x + 5 * y - 2, -3 * x + 6 * y - 15), (x, y))  
solution[x], solution[y]
```

```
Out[46]: (-3, 1)
```

```
In [47]: f = x ** 4 - 3 * x ** 2 + 1  
sym.factor(f)
```

```
Out[47]: (x**2 - x - 1)*(x**2 + x - 1)
```

```
In [48]: sym.satisfiable(x & y)
```

```
Out[48]: {x: True, y: True}
```

```
In [49]: sym.Matrix([[1, 0], [0, 1]])
```

```
Out[49]: Matrix([  
    [1, 0],  
    [0, 1]])
```

```
In [51]: x, y = sym.symbols('x, y')  
A = sym.Matrix([[1, x], [y, 1]])  
A
```

```
Out[51]: Matrix([  
    [1, x],  
    [y, 1]])
```

```
In [52]: A**2
```

```
Out[52]: Matrix([  
    [x*y + 1, 2*x],  
    [2*y, x*y + 1]])
```

Equation Solving

solveset()

solveset($x^{**} 4 - 1$, x) ={-1,1,-I,I}

Matrices

A={[1,2][2,1]} find A**2

Automata Based Programming Paradigm

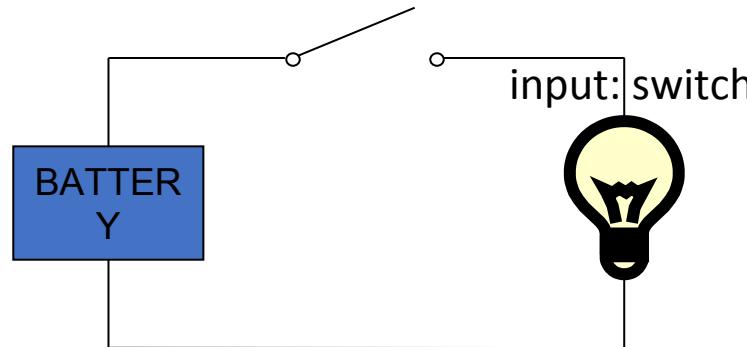
Introduction

Automata-based programming is a programming paradigm in which the program or its part is thought of as a model of a finite state machine or any other formal automation.

What is Automata Theory?

- Automata theory is the study of abstract computational devices
- Abstract devices are (simplified) models of real computations
- Computations happen everywhere: On your laptop, on your cell phone, in nature, ...

Example:



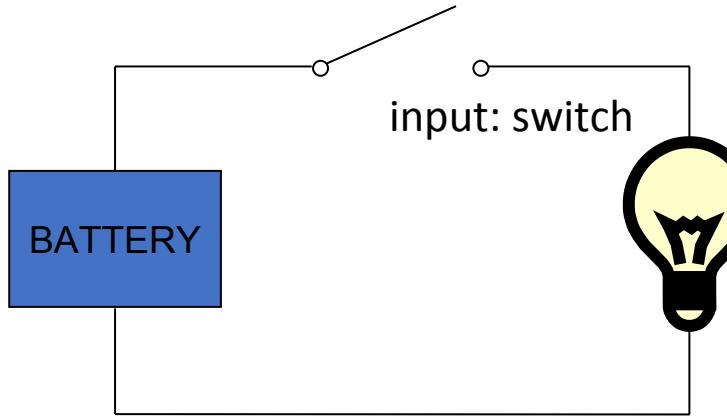
output: light bulb

actions: flip switch

states: on, off

Simple Computer

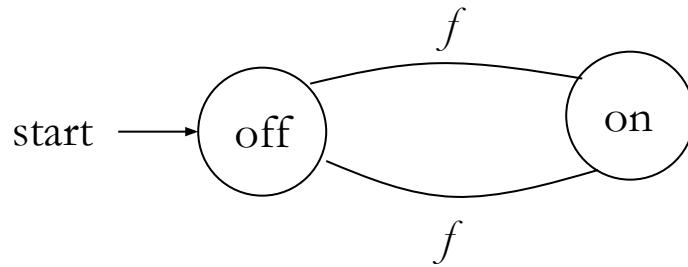
Example:



output: light bulb

actions: flip switch

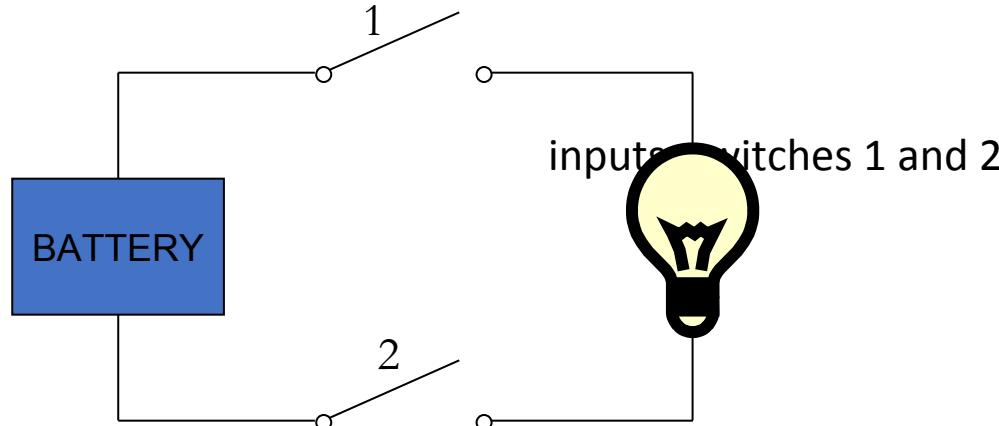
states: on, off



bulb is on if and only if there was an odd number of flips

Another “computer”

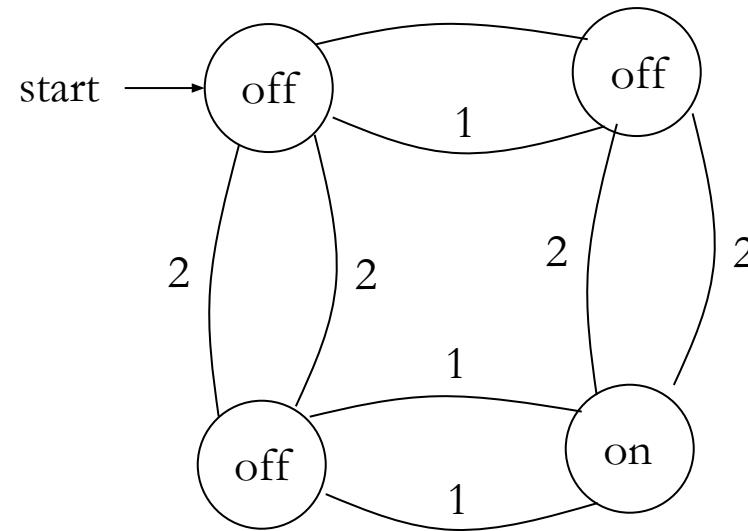
Example:



actions: 1 for “flip switch 1”

actions: 2 for “flip switch 2”

states: on, off



bulb is on if and only if both switches were flipped an odd number of times

Types of Automata

finite automata Devices with a finite amount of memory.
 Used to model “small” computers.

push-down
automata Devices with infinite memory that can be
 accessed in a restricted way.
 Used to model parsers, etc.

Turing
Machines Devices with infinite memory.
 Used to model any computer.

Alphabets and strings

A common way to talk about words, number, pairs of words, etc. is by representing them as strings

To define strings, we start with an alphabet

An **alphabet** is a finite set of symbols.

Examples:

$\Sigma_1 = \{a, b, c, d, \dots, z\}$: the set of letters in English

$\Sigma_2 = \{0, 1, \dots, 9\}$: the set of (base 10) digits

$\Sigma_3 = \{a, b, \dots, z, \#\}$: the set of letters plus the special symbol #

$\Sigma_4 = \{ (,) \}$: the set of open and closed brackets

Strings

A **string** over alphabet Σ is a finite sequence of symbols in Σ .

The empty string will be denoted by e

Examples:

abfbz is a string over $\Sigma_1 = \{a, b, c, d, \dots, z\}$

9021 is a string over $\Sigma_2 = \{0, 1, \dots, 9\}$

ab#bc is a string over $\Sigma_3 = \{a, b, \dots, z, \#\}$

))()() is a string over $\Sigma_4 = \{ (,) \}$

Languages

A **language** is a set of strings over an alphabet.

Languages can be used to describe problems with “yes/no” answers, for example:

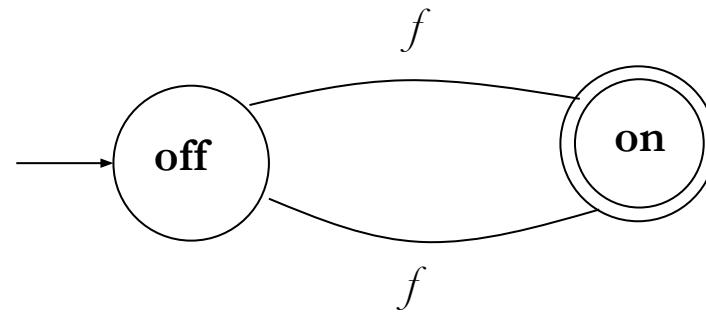
L_1 = The set of all strings over Σ_1 that contain the substring “SRM”

L_2 = The set of all strings over Σ_2 that are divisible by 7 = {7, 14, 21, ...}

L_3 = The set of all strings of the form s#s where s is any string over {a, b, ..., z}

L_4 = The set of all strings over Σ_4 where every (can be matched with a subsequent)

Finite Automata



There are states off and on, the automaton starts in off and tries to reach the “good state” on
What sequences of fs lead to the good state?

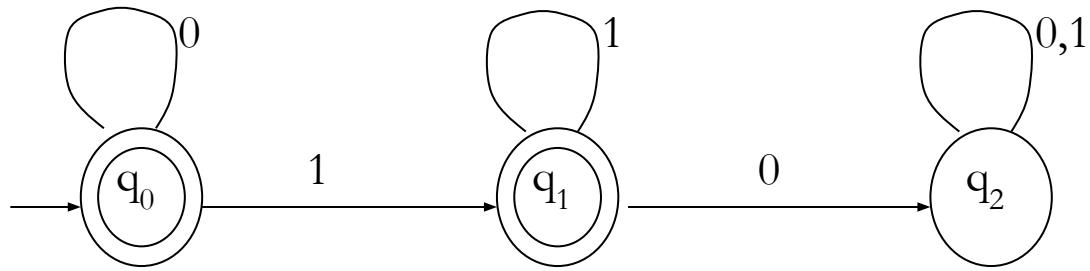
Answer: $\{f, fff, fffff, \dots\} = \{f^n : n \text{ is odd}\}$

This is an example of a deterministic finite automaton over alphabet $\{f\}$

Deterministic finite automata

- A deterministic finite automaton (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where
 - Q is a finite set of states
 - Σ is an alphabet
 - $\delta: Q \times \Sigma \rightarrow Q$ is a transition function
 - $q_0 \in Q$ is the initial state
 - $F \subseteq Q$ is a set of accepting states (or final states).
- In diagrams, the accepting states will be denoted by double loops

Example



alphabet $\Sigma = \{0, 1\}$

start state $Q = \{q_0, q_1, q_2\}$

initial state q_0

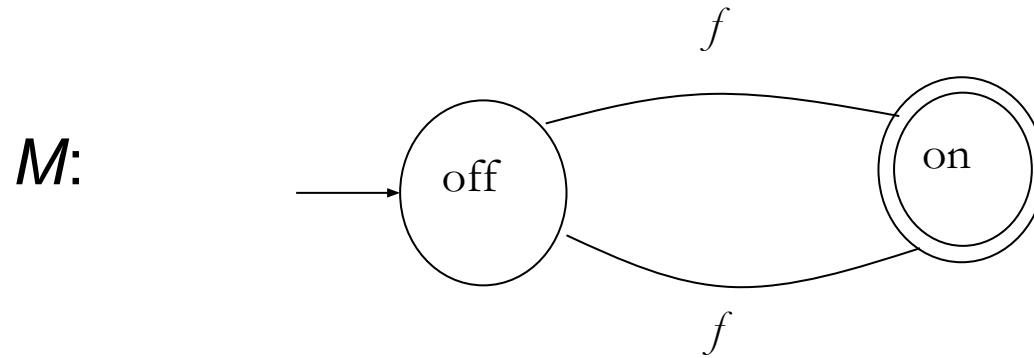
accepting states $F = \{q_0, q_1\}$

transition function δ :

inputs		
states	0	1
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_2	q_2

Language of a DFA

The **language of a DFA** $(Q, \Sigma, \delta, q_0, F)$ is the set of all strings over Σ that, starting from q_0 and following the transitions as the string is read left to right, will reach some accepting state.

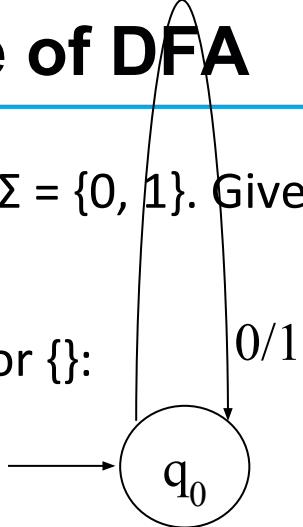


- Language of M is $\{f, fff, fffff, \dots\} = \{f^n : n \text{ is odd}\}$

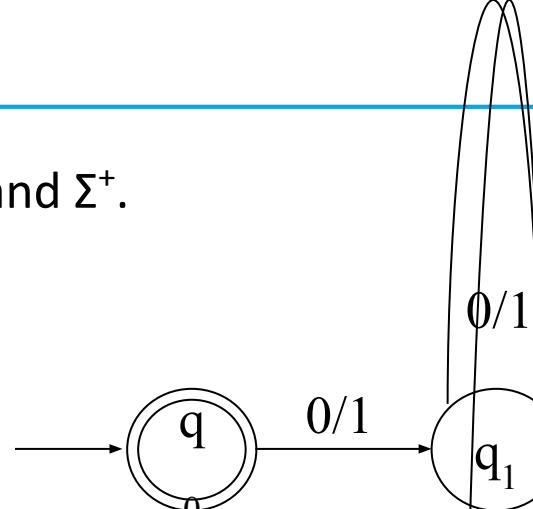
Example of DFA

1. Let $\Sigma = \{0, 1\}$. Give DFAs for $\{\}$, $\{\epsilon\}$, Σ^* , and Σ^+ .

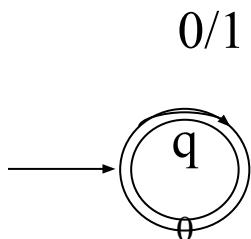
For $\{\}$:



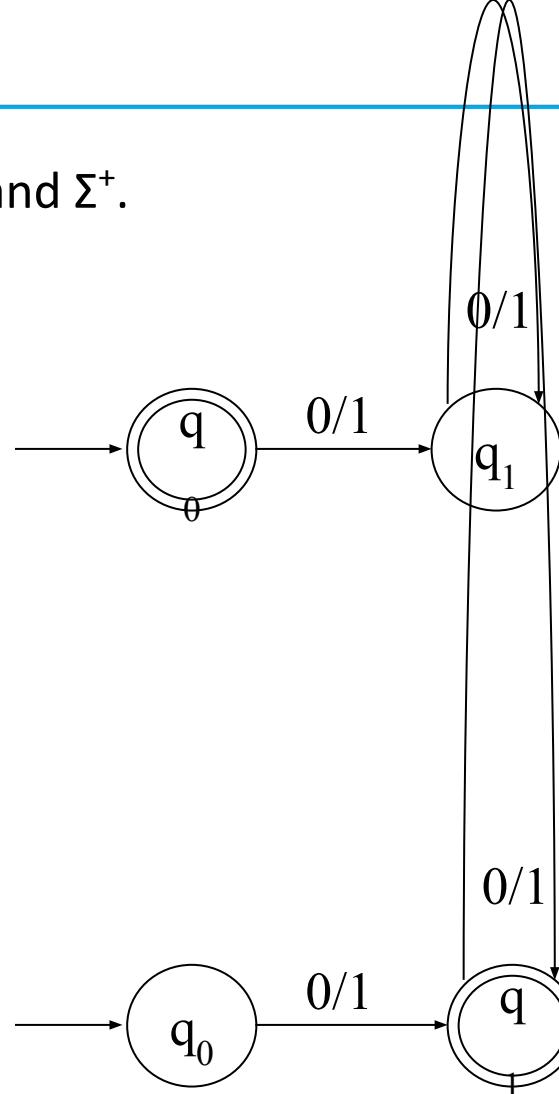
For $\{\epsilon\}$:



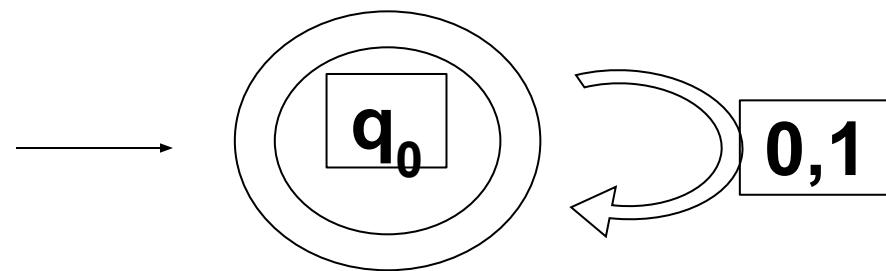
For Σ^* :



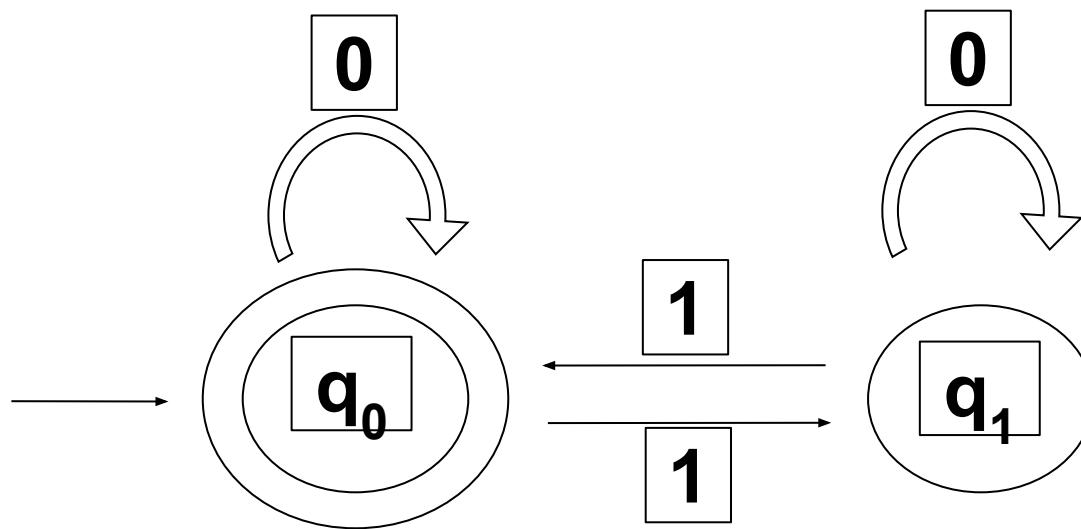
For Σ^+ :



Example of DFA



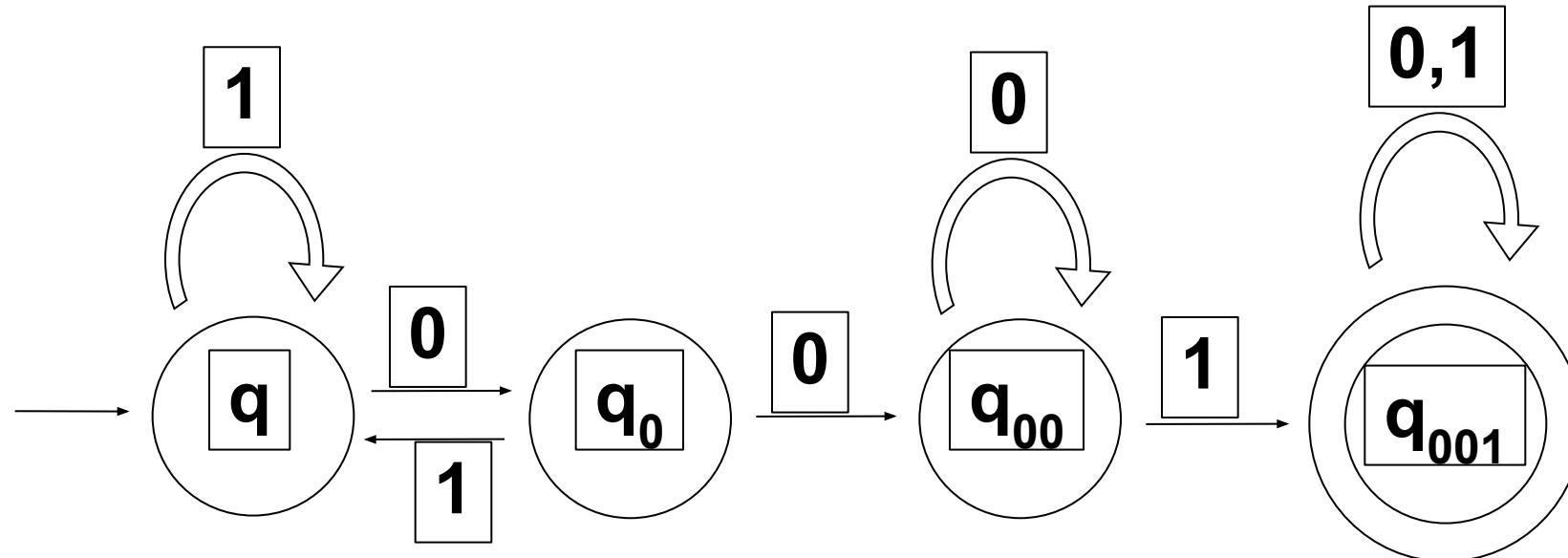
$$L(M) = \boxed{\{0,1\}^*}$$



$$L(M) = \boxed{\{ w \mid w \text{ has an even number of } 1s \}}$$

Example of DFA

Build an automaton that accepts all and only those strings that contain 001



Example of DFA using Python

```
from automata.fa.dfa import DFA
# DFA which matches all binary strings ending in an odd number of '1's
dfa = DFA(
    states={'q0', 'q1', 'q2'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q0', '1': 'q1'},
        'q1': {'0': 'q0', '1': 'q2'},
        'q2': {'0': 'q2', '1': 'q1'}
    },
    initial_state='q0',
    final_states={'q1'}
)
dfa.read_input('01') # answer is 'q1'
dfa.read_input('011') # answer is error
print(dfa.read_input_stepwise('011'))
Answer # yields:
# 'q0'  # 'q0'  # 'q1'
# 'q2'  # 'q1'
```

```
if dfa.accepts_input('011'):
    print('accepted')
else:
    print('rejected')
```

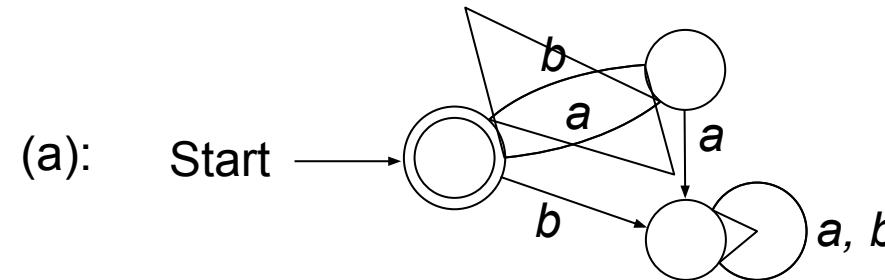
Questions for DFA

Find an DFA for each of the following languages over the alphabet $\{a, b\}$.

(a) $\{(ab)^n \mid n \in \mathbb{N}\}$, which has regular expression $(ab)^*$.

Solution

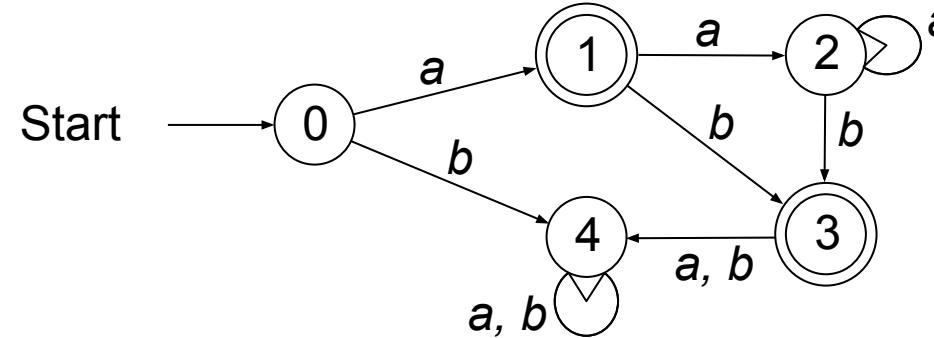
:



b) Find a DFA for the language of $a + aa^*b$.

Solution

:



Questions for DFA

c) A DFA that accepts all strings that contain 010 or do not contain 0.

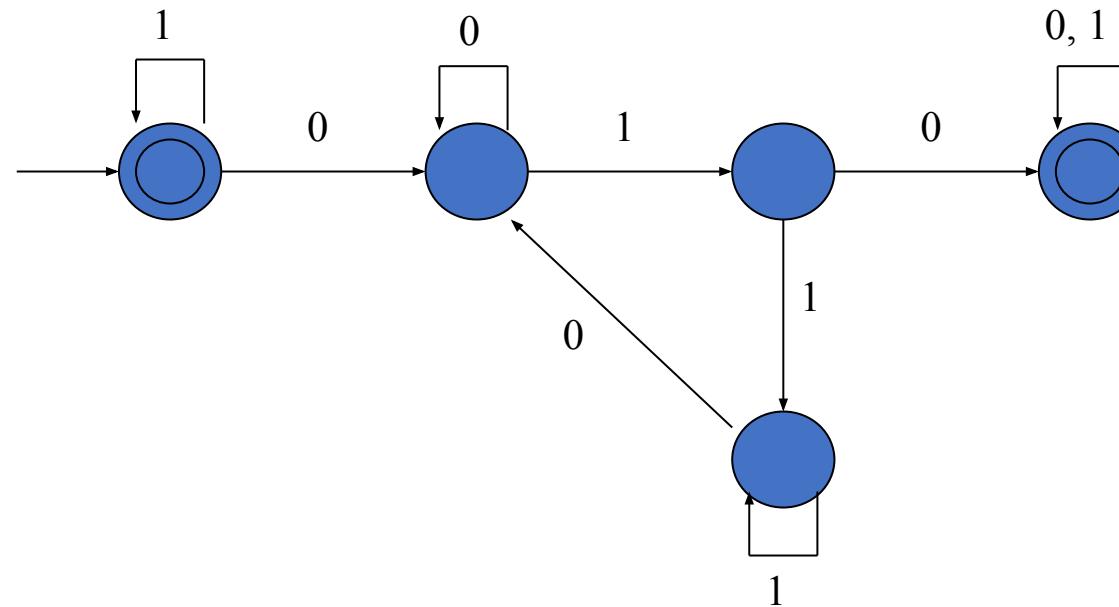
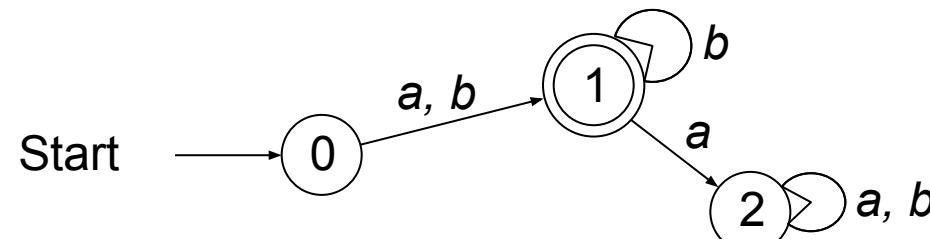


Table Representation of a DFA

A DFA over A can be represented by a transition function $T : \text{States} \times A \rightarrow \text{States}$, where $T(i, a)$ is the state reached from state i along the edge labelled a , and we mark the start and final states. For example, the following figures show a DFA and its transition table.



T	a	b
start	0	1 1
final	1	2 1
2	2	2

Sample Exercises - DFA

1. Write a automata code for the Language that accepts all and only those strings that contain 001
2. Write a automata code for $L(M) = \{ w \mid w \text{ has an even number of } 1s\}$
3. Write a automata code for $L(M) = \{0,1\}^*$
4. Write a automata code for $L(M) = a + aa^*b$.
5. Write a automata code for $L(M) = \{(ab)^n \mid n \in \mathbb{N}\}$
6. Write a automata code for Let $\Sigma = \{0, 1\}$.

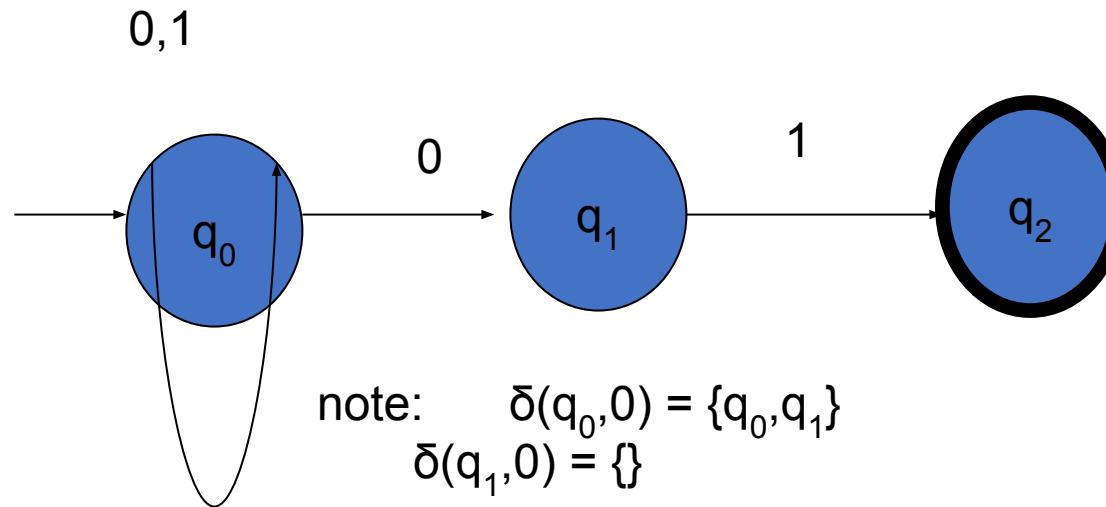
Given DFAs for $\{\}$, $\{\epsilon\}$, Σ^* , and Σ^+ .

NDFA

- A nondeterministic finite automaton M is a five-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where:
 - Q is a finite set of states of M
 - Σ is the finite input alphabet of M
 - $\delta: Q \times \Sigma \rightarrow \text{power set of } Q$, is the state transition function mapping a state-symbol pair to a subset of Q
 - q_0 is the start state of M
 - $F \subseteq Q$ is the set of accepting states or final states of M

Example NDFA

- NFA that recognizes the language of strings that end in 01

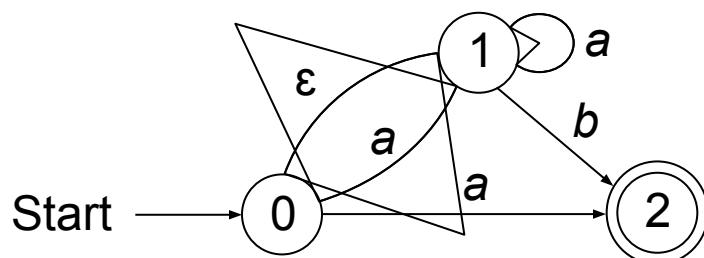


Exercise: Draw the complete transition table for this NFA

NDFA

A nondeterministic finite automaton (NFA) over an alphabet A is similar to a DFA except that epsilon-edges are allowed, there is no requirement to emit edges from a state, and multiple edges with the same letter can be emitted from a state.

Example. The following NFA recognizes the language of $a + aa^*b + a^*b$.



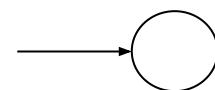
	T	a	b	ϵ
start	0	{1, 2}	\emptyset	{1}
	1	{1}	{2}	\emptyset
final	2	\emptyset	\emptyset	\emptyset

Table representation of NFA

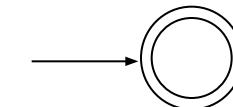
An NFA over A can be represented by a function $T : \text{States} \times A \cup \{\lambda\} \rightarrow \text{power}(\text{States})$, where $T(i, a)$ is the set of states reached from state i along the edge labeled a , and we mark the start and final states. The following figure shows the table for the preceding NFA.

Examples

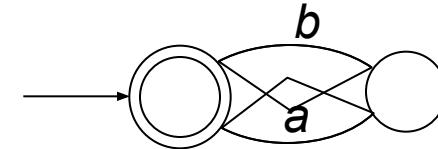
Solutions: (a): Start



(b): Start

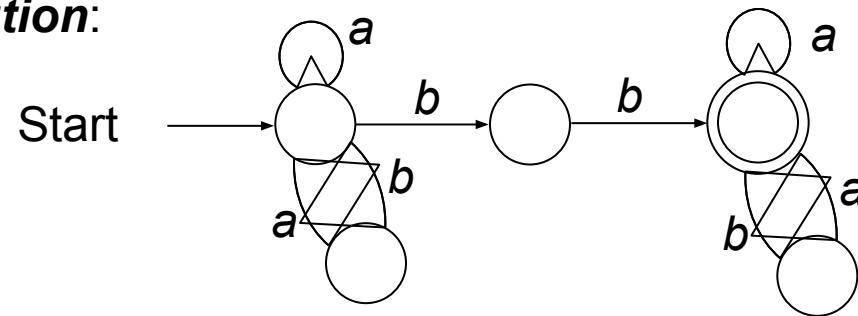


(c): Start



Find an NFA to recognize the language $(a + ba)^*bb(a + ab)^*$.

A solution:



Examples

Algorithm: *Transform a Regular Expression into a Finite Automaton*

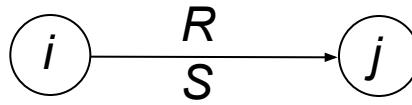
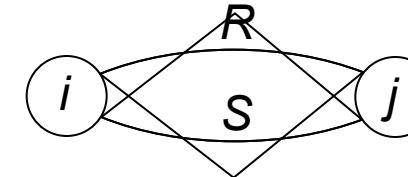
Start by placing the regular expression on the edge between a start and final state:



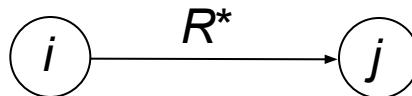
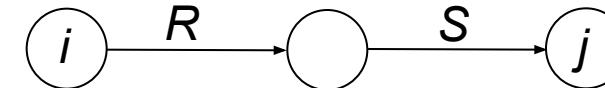
Apply the following rules to obtain a finite automaton after erasing any \emptyset -edges.



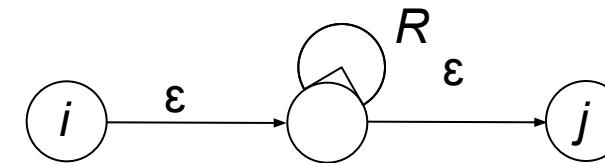
transforms
to



transforms
to

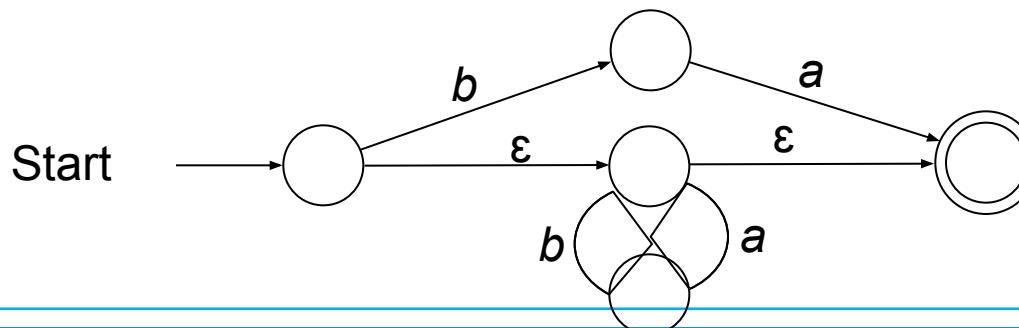


transforms
to



Quiz. Use the algorithm to construct a finite automaton for $(ab)^* + ba$.

Answer:



Example of NFA using Python

```
from automata.fa.nfa import NFA
# NFA which matches strings beginning with 'a', ending with 'a', and
containing
# no consecutive 'b's
nfa = NFA(
    states={'q0', 'q1', 'q2'},
    input_symbols={'a', 'b'},
    transitions={
        'q0': {'a': {'q1'}},
        # Use " as the key name for empty string (lambda/epsilon)
transitions
        'q1': {'a': {'q1'}, '': {'q2'}},
        'q2': {'b': {'q0'}}
    },
    initial_state='q0',
    final_states={'q1'}
)
```

```
nfa.read_input('aba')
ANSWER :{'q1', 'q2'}
```



```
nfa.read_input('abba')
ANSWER: ERROR
```



```
nfa.read_input_stepwise('aba')
if nfa.accepts_input('aba'):
    print('accepted')
else:
    print('rejected')
ANSWER: ACCEPTED
nfa.validate()
ANSWER: TRUE
```

Sample Exercises - NFA

1. Write a automata code for the Language that accepts all end with 01
2. Write a automata code for $L(M) = a + aa^*b + a^*b$.
3. Write a automata code for Let $\Sigma = \{0, 1\}$.

Given NFAs for $\{\}$, $\{\epsilon\}$, $\{(ab)^n \mid n \in \mathbb{N}\}$, which has regular expression $(ab)^*$.

Dependent Types Paradigms

Dependent Types Paradigms

Unit-IV (15 Session)

Session 6-10 cover the following topics:-

- *Dependent Type Programming Paradigm*- S6-SLO1
- *Logic Quantifier: for all, there exists*- S6-SLO2
- *Dependent functions, dependent pairs*– S7-SLO 1
- *Relation between data and its computation*– S7-SLO 2
- *Other Languages: Idris, Agda, Coq* S8-SLO 1
- Demo: Dependent Type Programming in Python S8-SLO2

Lab 11: Dependent Programming (Case Study) (S8)

Assignment : Comparative study of Dependent programming in Idris, Agda, Coq

TextBook:

- 1) Amit Saha, Doing Math with Python: Use Programming to Explore Algebra, Statistics,Calculus and More, Kindle Edition, 2015

URL :

- <https://tech.peoplefund.co.kr/2018/11/28/programming-paradigm-and-python-eng.html>
- <https://freecontent.manning.com/a-first-example-of-dependent-data-types/>

Introduction

- A constant problem:
- Writing a correct computer program is hard
- Proving that a program is correct is even harder
- Dependent Types allow us to write programs and know they are correct before running them.

What is correctness?

- What does it mean to be “correct”?
- Depends on the application domain, but could mean one or more of:
 - **Functionally correct** (e.g. arithmetic operations on a CPU)
 - **Resource safe** (e.g. runs within memory bounds, no memory leaks, no accessing unallocated memory, no deadlock. . .)
 - **Secure** (e.g. not allowing access to another user’s data)

What is Type?

- In **programming**, types are a means of classifying values
- Exp: values 94, "thing", and [1,2,3,4,5] classified as an integer, a string, and a list of integers
- For a ***machine***, types describe how bit patterns in memory are to be interpreted.
- For a ***compiler or interpreter***, types help ensure that bit patterns are interpreted consistently when a program runs.
- For a ***programmer***, types help name and organize concepts, aiding documentation and supporting interactive editing environments.

Introductions

- In computer science and logic, a dependent type is a type whose definition depends on a value.
- It is an overlapping feature of type theory and type systems.
- Used to encode logic's quantifiers like "for all" and "there exists".
- Dependent types may help reduce bugs by enabling the programmer to assign types that further restrain the set of possible implementations.
- Exp: Agda, ATS, Coq, F*, Epigram, and Idris

Dependent Type Example

- **Exp matrix arithmetic**
- **Matrix type** - □ refined it to include the number of rows and columns.
- Matrix 3 4 is the type of 3×4 matrices.
- In this type, 3 and 4 are ordinary values.
- A *dependent type*, such as Matrix, is a type that's calculated from some other values.
- In other words, it *depends on* other values.
- **Definition**
 - A data type is a type which is computed from a *dependent* other value.

Elements of dependent types

- **Dependent functions**
 - The return type of a dependent function may depend on the *value* (not just type) of one of its arguments
 - For instance, a function that takes a positive integer n may return an array of length n , where the array length is part of the type of the array.
 - (Note that this is different from [polymorphism](#) and [generic programming](#), both of which include the type as an argument.)
- **Dependent pairs**
 - A dependent pair may have a second value of which the type depends on the first value

Formal definition

Π type [edit]

Loosely speaking, dependent types are similar to the type of an indexed family of sets. More formally, given a type $A : \mathcal{U}$ in a universe of types \mathcal{U} , one may have a **family of types** $B : A \rightarrow \mathcal{U}$, which assigns to each term $a : A$ a type $B(a) : \mathcal{U}$. We say that the type $B(a)$ varies with a .

A function whose type of return value varies with its argument (i.e. there is no fixed codomain) is a **dependent function** and the type of this function is called **dependent product type**, **pi-type** or **dependent function type**.^[3] For this example, the dependent function type is typically written as

$$\prod_{x:A} B(x)$$

or

$$\prod_{x:A} B(x).$$

If $B : A \rightarrow \mathcal{U}$ is a constant function, the corresponding dependent product type is equivalent to an ordinary **function type**. That is, $\prod_{x:A} B$ is judgmentally equal to $A \rightarrow B$ when B does not depend on x .

Formal definition

Σ type [edit]

The dual of the dependent product type is the **dependent pair type**, **dependent sum type**, **sigma-type**, or (confusingly) **dependent product type**.^[3] Sigma-types can also be understood as **existential quantifiers**. Continuing the above example, if, in the universe of types \mathcal{U} , there is a type $A : \mathcal{U}$ and a family of types $B : A \rightarrow \mathcal{U}$, then there is a dependent pair type

$$\sum_{x:A} B(x).$$

The dependent pair type captures the idea of an ordered pair where the type of the second term is dependent on the value of the first. If

$$(a, b) : \sum_{x:A} B(x),$$

then $a : A$ and $b : B(a)$. If B is a constant function, then the dependent pair type becomes (is judgementally equal to) the **product type**, that is, an ordinary Cartesian product $A \times B$.

Pseudo-code

- **General Code**

```
float myDivide(float a, float b)
{ if (b == 0)
return ???;
Else
return a / b;
}
```

- **Dependent Type Code**

```
float myDivide3
(float a, float b, proof(b != 0)
p)
{
return a / b;
}
```

Auto Checking done here

Python Simple Example

```
from typing import Union def return_int_or_str(flag: bool) ->
    Union[str, int]:
    if flag:
        return 'I am a string!'
    return 0
```

Dependent Type

- » pip install mypy typing_extensions
- from typing import overload
- from typing_extensions import Literal

Literal

Literal type represents a specific value of the specific type.

```
from typing_extensions import Literal
def function(x: Literal[1]) -> Literal[1]:
    return x
function(1)
# => OK!
function(2)
# => Argument has incompatible type "Literal[2]"; expected "Literal[1]"
```

Python Example

x :: Iterator[T1]

y :: Iterator[T2]

z :: Iterator[T3]

product(x, y) :: Iterator[Tuple[T1, T2]]

product(x, y, z) :: Iterator[Tuple[T1, T2, T3]]

product(x, x, x, x) :: Iterator[Tuple[T1, T1, T1, T1]]

- All the above replaced by
 - def product(*args :Tuple[n]) -> Iterator[Tuple[n]]: pass

A first example: classifying vehicles by power source IDRIS Example

Listing 1 Defining a dependent type for vehicles, with their power source in the type (vehicle.idr)

```
data PowerSource = Petrol | Pedal          ①  
data Vehicle : PowerSource -> Type where  
  Bicycle : Vehicle Pedal                ②  
  Car : (fuel : Nat) -> Vehicle Petrol  ③  
  Bus : (fuel : Nat) -> Vehicle Petrol   ④
```

- ① An enumeration type describing possible power sources for a vehicle
- ② A Vehicle's type is annotated with its power source
- ③ A vehicle powered by pedal
- ④ A vehicle powered by petrol, with a field for current fuel stocks

IDRIS Second Example

Listing 2 Reading and updating properties of Vehicle

wheels : Vehicle power -> Nat ①

wheels Bicycle = 2 wheels (Car fuel) = 4 wheels (Bus fuel) = 4

refuel : Vehicle Petrol -> Vehicle Petrol ②

refuel (Car fuel) = Car 100 refuel (Bus fuel) = Bus 200

① Use a type variable, power, because this function works for all possible vehicle types.

② Refueling only makes sense for vehicles that carry fuel.
Restrict the input and output type to Vehicle Petrol.

References

- <http://www.cs.ru.nl/dtp11/slides/brady.pdf>
- <https://freecontent.manning.com/a-first-example-of-dependent-data-types/>
- https://en.wikipedia.org/wiki/Dependent_type
- <https://livebook.manning.com/book/type-driven-development-with-idris/chapter-1/13>
- <https://github.com/python/mypy/issues/366>