AlgoHack : පයිතන් #4

දත්ත ආකෘති - ශබ්දකෝෂ - ගොනු - වස්තු

කතුවරු
නිරංජන් මීගම්මන, නවීන් තමිදු

සංස්කරණය
විශ්ව කුමාර

AlgoHack කෘඩා අවදියේදී ළමුනට පරිගණක විද්‍යාව සහ ක්‍රම ලේඛනය ඉගැන්වීමට ශිල්ප සයුර කල ආරම්භයකි. එයට Google for Education සහ ශ්‍රී ලංකා පරිගණක සංගමය සහාය ලබා දේ

1.0 ගොඩගැසුම් ආකෘතිය (stack)

මෙය මූලික ක්‍රියා දෙකක් ඇති, එකිනෙක මත දත්ත ගොඩගසන ලැයිස්තු ආකෘතියකි.

push : ආකෘතිය මුලට දත්තයක් ඇතුළ් කරයි.
pop: අවසානයට ඇතුලත් කළ දත්තය ඉවත්කර ප්‍රතිදානය කරයි.
මේ සඳහා අවසානයට ආ අය පළමුව ඉවත්කරන LIFO (Last In First Out) ක්‍රමය භාවිතා වේ.
එය හරියට එක මත එක තබන ගඩොල් ගොඩකට සමානවේ. ගඩොල් එකතු කරන්නේ සහ ඉවත්කරන්නේ උඩිනි.

```
stack= [] # හිස් ගොඩගැසුම
stack.append(1) # අගයක් එක් කරන්න
stack.append("Sri Lanka")
stack.append( 4)
stack.append( "India")
print(stack)
a=stack.pop() # පළමු අගය ඉවතට
print(a)
print(stack)  # ඉතිරි ලැයිස්තුව
stack.pop()   # පළමු අගය ඉවතට
```

```
print(stack)  # ඉතිරි ලැයිස්තුව
```

ප්‍රතිදාන පහත පරිදි වේ

```
[1, 'Sri Lanka', 4, 'India']
India
[1, 'Sri Lanka', 4]
[1, 'Sri Lanka']
```

2.0 පොලිම් අංකනීය (Queue)

පොලිම් අංකනීය පළමුව ආ අය පළමුව ඉවත්වන FIFO (First In First Out) ක්‍රමවේදය භාවිතා කරයි
පෝලිම ක්‍රමවේද 2 ක් සහිත දත්ත අංකනීයකි.
push : පෝලිම අගට අගයක් අැතුලත් කරයි (enque)
pop: පෝලිමේ මුලම අගය ඉවත් කරයි (deque)

```
queue= []
queue.append(1)
queue.append("Sri Lanka")
queue.append( 4)
queue.append( "India")
print(queue)

x=queue.pop(0)
print(x)
print(queue)
y=queue.pop(0)
print(y)
print(queue)
```

පහත ප්‍රතිදානය නිරීක්ෂණය කරන්න

```
[1, 'Sri Lanka', 4, 'India']
1
['Sri Lanka', 4, 'India']
Sri Lanka
[4, 'India']
```

3.0 ලැයිස්තුවක මාන ප්‍රකාශ කිරීම

ලැයිස්තුවක භාවිතයට එහි අවයව ගණන ප්‍රකාශ කළ යුතුය. දැනට පවතින අවයවවල අගයන් අපට වෙනස් කළ හැක. නමුත් දැනට නොපවතින අවයව වලට අගයන් අනුයුක්ත කළ නොහැක.

```
myList=[] # හිස් අරාව
myList[0]=7 # ක්‍රමලේඛය ක්‍රියා නොකරනු ඇත
myList=[2] #  මාන කළ අරාව
myList[0]=7
myList[1]=8
myList.append(7)

සංඛ්‍යා 100 ක ලයිස්තුවක් නිර්මාණයට කෝඩ්යක් ලියන්න
myList=[]
for i in range(0,100):
    myList.append(i)

print (myList)

# අවයව 10 ක් සහිත ලයිස්තුවක් ප්‍රකාශ කිරීම
# මෙහිදී ලැයිස්තුව නිර්මාණය වන්නේ [ ] තුල ඇති චක්‍රයෙනි.
```

```
n=10
myList=[0 for i in range(n)]
print (myList) # අවයවවල අගයන් 0 ලෙස ප්‍රකාශ කරයි
```

ප්‍රතිදානය වන්නේ
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

තවත් ප්‍රතිදානයක්

```
myList=[i*i for i in range(10)]
print (myList)
```

myList නම් ලයිස්තුව
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81] ට හරවයි

අරාවක් නිර්මාණය කිරීමට ශ්‍රිතයක් භාවිතා කිරීම

```
def dim3( n):
  return n*n*n

print (dim3(2))
myList=[dim3(i) for i in range(10)]
print (myList)
```

4.0 ශබ්දකෝෂ

පිළිවෙලට සැකසීම, අපේක්ෂිත හැරවීම හා ප්‍රමාණය ගණන් කිරීම.
ශබ්දකෝෂ (Dictionaries) වලදී අගය යුගල් දෙනින් (:) මගින් වෙන් කර තබාගනී.

අවයව වෙන් කර තබා ගැනීමේදී කොමා භාවිතා කරන අතර අවයව ගොනු සඟල වරහන් මගින් වෙන් කරයි ({ })

#ශබ්දකෝෂ ප්‍රකාශ කිරීම
dict = {'Name': 'Ganesh', 'Age': 17, 'Class': 12}
print (dict['Name'])
print (dict['Age'])
ප්‍රතිදානය පහත ලෙස වේ

Ganesh
17

dict['Age'] = 18 # දත්තය යාවත්කාලීන කරයි.
dict['Class'] = 13
dict['School'] = "Ranabima" #අලුත් දත්තය අැතුලත් කරයි
print(dict)
ප්‍රතිදානය කුමක් විය හැකිද**?**

ශබ්දකෝෂ ක්‍රම
dict.clear - ශබ්දකෝෂයේ සියලු අවයව මකා දමයි
dict.items - අවයව ටියුපලයක් ලබාදේ
dict.keys - අවයව වල යතුරු ලබාදේ
dict.values- අවයව වල සියළු අගයන් ලබාදේ
dict.pop(key) - යතුර අනුව අවයවද ඉවත් කර ලබාදේ
dict.get(key) - යතුර අනුව අවයවද අවයවයක් සොයා ගනී
dict.update(new) - දත්ත තවත් ශබ්දකෝෂයකින් අළුත් කරයි.
dict.copy - ශබ්දකෝෂයක් පිටපත් කරයි
dict.setdefault(key) - යතුරක් මගින් අවයව සොයා ගනී
dict. has_key(key) - දී අැති යතුර ශබ්දකෝෂයක අඩංගුද බලයි

ප්‍රතිදාන කුමක් වේද**?**

```
dict = {'Name': 'Ganesh', 'Age': 17, 'Class': 12}
print (dict['Name']) # ප්‍රතිදානය Ganesh වේ
del dict['Name']
print (dict)
x=dict.get('Age')
print(x)
dict.clear()
print (dict)
del dict
print(dict)
```

```
dict = {'Name': 'Ganesh', 'Age': 17, 'Class': 12}
x=dict.keys()
print (x) # ['Name', 'Age', 'Class'] ප්‍රතිදානය කරයි

y=dict.values()
print (y) # ['Ganesh', 17, 12] ප්‍රතිදානය කරයි

z=dict.items()
print (z) # [('Name', 'Ganesh'), ('Age', 17), ('Class', 12)]
```

```
dict = {'Name': 'Ganesh', 'Age': 17, 'Class': 12}
x=dict.keys()
print (x) # ['Name', 'Age', 'Class'] ප්‍රතිදානය කරයි

y=dict.values()
print (y)
# ['Ganesh', 17, 12] ප්‍රතිදානය කරයි
```

```
z=dict.items()
print (z)
```
[('Name', 'Ganesh'), ('Age', 17), ('Class', 12)] ප්‍රතිදානය කරයි

```
dict1 = {'Name': 'Ganesh', 'Age': 17}
dict2 = dict1.copy()
print (dict2)
```

ශබ්දකොෝෂයේ යතුර අඩංගු නම් සහායය ලෙස ප්‍රතිදානය වේ

```
x=dict.has_key('Age')
print(x) # ප්‍රතිදානය සහායය වේ
```

```
dict2 = dict1.copy()
print (dict2)
del dict2['Age'] # යතුරුනාමය ඉවත් කරයි
print (dict2)
print (dict1)
```

5. ගොනු (Files) හැසිරවීම

ගොනුවක් පරිගණකයේ බයිට ලැයිස්තුවක් ලෙස සැලකිය හැක. ගොනුවක දත්ත කියවීමට එය විවර කළ යුතු අතර, දත්ත ලියන ලද්දේ නම් භාවිතයෙන් පසු වසා දැමිය යුතුවේ.

```
fo = open("myfile.txt") # ගොනුව විවෘත කරයි
print (fo.name) # ගොනුවේ නාමය ප්‍රතිදානය කරයි
```

ගොනුවක් යාවත් කාලීන කිරීම.
ගොනුවක් වවන හෝ ද්වීමය වෙසින් විවර කළ හැකිය.
```
# wb යන්න ද්වීමය වෙසින් ලිවීමක් බව අදහස් කරයි
fo = open("myfile.txt", "wb")
```

```
fo.write( "Python is cool\n") #ගොනුවට ලියයි
fo.close() # ගොනුව වසා දමයි
```

ගොනු විවර කරන වෙස් (modes)

r කියවීම පමණයි (read only)
rb ද්වීමය වෙසින් කියවීම පමණයි (read only in binary)
r+ කියවීමට හා ලිවීමට (reading and writing)
b+ ද්වීමය වෙසින් කියවීමට හා ලිවීමට
w ලිවීමට පමණයි (writing only)
wb ද්වීමය වෙසින් ලිවීමට පමණයි
w+ ලිවීමට හා කියවීමට
wb+ ද්වීමය වෙසින් ලිවීමට හා කියවීමට
a අගට එකතු කිරීමට (appending)
ab ද්වීමය වෙසින් අගට එකතු කිරීමට
a+ අගට එකතු කිරීමට හා කියවීමට
ab+ ද්වීමය වෙසින් අගට එකතු කිරීමට හා කියවීමට

විවර කරන ලද ගොනුවක වචන කියවීම

```
fo = open("myfile.txt", "r+")
str = fo.read() # ගොනුවේ අවසානය දක්වා කියවන්න
print (str) # ප්‍රතිදානය
fo.close() # ගොනුව වසන්න

fo = open("myfile.txt", "r+") #
str = fo.read(6) # අකුරු6 ක් කියවන්න
print (str)
fo.close()
```

ගොනුවක් විවර කර කියවීමේදී යොමු තුඩක් (Pointer) භාවිතා කරයි. යොමුතුඩ ගොනුවේ ස්ථාන ගැන්වීමෙන්, එතැන් සිට දත්ත කියවිය හැක.

```
fo = open("myfile.txt", "r+")
str = fo.read(6)
print (str)
position = fo.tell()
print (position) # වත්මන් පිහිටුම ලබාදේ
str = fo.read(3)
print (str)
position = fo.tell()
print (position)

# යොමුතුඩ ස්ථාන ගත කිරීම
position = fo.seek(0, 0)
str = fo.read(6)
print (str)
fo.close()
```

පෙළ් කියවීම

```
fo = open("myfile.txt", "rw+")
line = fo.readline() //පේලියක් කියවන්න
print (line)
line = fo.readline(5) # ඊළඟ පෙළියෙන් අකුරු5 ක් කියවයි
print (line)
fo.close()
```

ගොනුවේ අවසානය දක්වා සියලු පේළි අරාවකට කියවන්න

```
fo = open("myfile.txt", "rw+")
lines = fo.readlines()
print (lines)
fo.close()
```

ගොනු මෙහෙයම සඳහා **os** මොඩියුලය ආනයනය

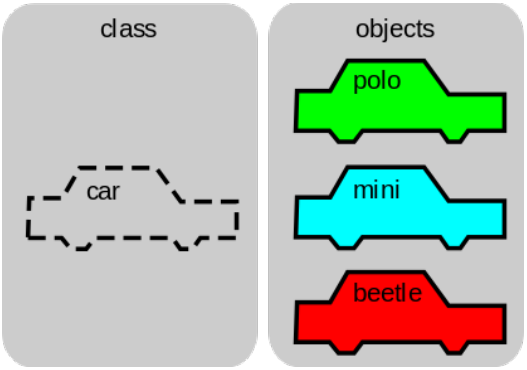
```
import os
myfile.txt ගොනුව myfile2.txt සේ නම් කරයි
os.rename( " myfile.txt", "myfile2.txt" )
os.remove("myfile.txt") # ගොනුව මකා දමයි
os.mkdir("newdir") # "newdir" ෆෝල්ඩරය සාදයි
os.chdir("/home/newdir") # වත්මන් ෆෝල්ඩරය කරයි
os.getcwd()  # වැඩකරන ෆෝල්ඩරයට එයි
os.rmdir( "/home/newdir" ) # ෆෝල්ඩරය මකා දමයි
```

6. වස්තුගත ක්‍රමලේඛණය Object Oriented programming

අප මෙතෙක් ශ්‍රීත ආශ්‍රිතව දත්ත හැසිරවූයෙමු. වස්තුගත ක්‍රමලේඛනයේදී අපි දත්ත හා ශ්‍රීත කේත වස්තුවක් තුලට කැටිකර භාවිතා කරමු. ඉතා සංකීර්ණ හා විශාල මෘදුකාංග නිශ්පාදනයේදී වස්තුගත ක්‍රමලේකනය භාවිතා කරයි.

පංති (Classes) සහ වස්තු (objects) වස්තුගත ක්‍රමලේකනයේ මූලිකාංග වේ. පංතියක අවස්ථාවක් (instance) වස්තුවක් වේ. එක පංතියක අඩංගු විචල්‍ය හා ශ්‍රීත එම පංතියෙන් වස්තුවක් නිර්මාණය කරන විට අැතුලත් වේ. එම ක්‍රමලේඛනයක එකම පංතියකට අයත් වස්තු ගණනාක් නිර්මාණය කර භාවිතා කල හැකි අතර එම වස්තුවල හැසිරීම එක හා සමාන වේ.

මිනිසා පංතියක් නම්. විශ්ව වස්තුවකි. නිරංජන් තවත් වස්තුවකි. වස්තු දෙකටම අැවිදීම, දිවීම, නිදාගැනීම යන ශ්‍රීත පොදුවේ. වස්තු දෙකේම නම, උස, බර යන ලක්ෂණ අඩංගු මුත් ඒවා වෙනස් විය හැක.



කාර් පංතියකින් වස්තු ගණනාක් නිර්මාණය කල හැක.

පංතියක් නිර්මාණය කරන්නේ **class** පදයෙනි.

ඔබට person නම් පංතියක් ඇත. එම පංතියේ පවතින අවස්ථාවක් හෝ නිදර්ශණයක් Vishva ලෙස නම් කරමු.

```
class Person:
    pass  # කිසිවක් නොකරයි

Vishva = Person()  # Vishva වස්තුව වේ
```

එය තවත් විදහාලමු

```
class Person:
    def sayhello(self):
        print('Hello, how are you?')

Vishva = Person()
```

Vishva.sayhello()

එය Person().sayhello() ලෙසද ලිවිය හැක.

sayhello() ශ්‍රිතය කිසිදු පරාමිතියක් (parameters) නොගනී නමුත් self නම් පරාමිතිය ස්වයංව ඇතුලත් වේ. එය එම වස්තුවයි.

__init__ ක්‍රමය (method)

මෙය විශේෂ python පංතියකි. එය පංතියකින් වස්තුවක් නිර්මාණය කිරීමේදී ආරම්භක අගයන් ලබාදීමට භාවිතා වේ.

පහත උදාහරණයේදී වස්තුව නිර්මාණය කරන අවස්ථාවේ ලබාදෙන name පරාමිතිය එම වස්තුවේ නමට අනුයුක්ත කෙරේ. සමස්ථ කේතනය තුල එම වස්තුව හඳුනාගන්නේ එම නමිනි.

```
class Person:
    def __init__(self, name):
        self.name = name නම අනුයුක්තය

    def sayhello(self):
        print('Hello, my name is', self.name)

p = Person('Vishva')
p.sayhello()
```

ප්‍රතිදානය
Hello, my name is Vishva

පංති හා වස්තු විචල්‍ය Class And Object Variables

වස්තුවට අයත් දත්ත සාමාන්‍ය විචල්‍යන් වන අතර ඒවා එම පංතියට හා වස්තුවට පමණක් අදාල වේ. එයට name spaces කියමු. ඒවා වර්ග දෙකකි.

පංති විචල්‍ය class variables
වස්තු විචල්‍ය object variable

පංති විචල්‍ය වස්තුවලට පොදු වේ. පංතියේ සියලු වස්තුවලට ඒවා භාවිතා කල හැක. පංති විචල්‍යයක අගය වස්තුවක් මගින් වෙනස් කලේ නම්, එම පංතිය නිර්මාණය කල සියලු වස්තු එම වෙනස දකී.

වස්තු විචල්‍ය එම වස්තුවට පමණක් අදාල වන අතර එකම පංතියක වුව අනිකුත් වස්තුවලට නොපෙනේ.

ඔබගේ පංතියේ සෑම සිසුවකුම 10 වන ශ්‍රේණියේ වන අතර, ඔබගේ උපන් දිනය වෙනස්ය. එය ඔබට ආවේණිකය

```
class Robot:
    population = 0 # රොබෝවරු ගණන - පංති විචල්‍ය

    def __init__(self, name): # ආරම්භක ශ්‍රිතය
        self.name = name
        print(self.name. “රොබෝ සූදානම් )
        #රොබෝවරයා නිර්මාණයේදී රොබෝගහණය වැඩිකෙරේ
        Robot.population += 1

    def discard(self):   # අඛලි ශ්‍රිතය
        print(self.name)
        print("අඛලි කරමින් පවතී")
        Robot.population -= 1
```

```
        if Robot.population == 0:
            print("අවසාන රොබෝය")
            print(self.name)
        else:
            print("තව රොබෝවරු සිටිති")
            print(Robot.population)

    def sayhello(self): # හලෝ ශ්‍රිතය
        print("ආයුබෝවන් මගේ නම")
        print(self.name)

    @classmethod
    def howmany(cls):
       print("අපට සිටින රොබෝ ගණන")
       (cls.population)

droid1 = Robot("R2-D2")
droid1.sayhello()
Robot.how_many()

droid2 = Robot("C-3PO")
droid2.sayhello()
Robot.howmany()

print("\රොබෝට වැඩ ඇත.\n")

print("රොබෝ වැඩ අවසානයි. අබලි කරන්න")
droid1.discard()
droid2.discard()
Robot.how_many()
```

ප්‍රතිදානය

R2-D2 රොබෝ සූදානම්
ආයුබෝවන් මගේ නම R2-D2.
අපට සිටින රොබෝ ගණන 1
C-3PO රොබෝ සූදානම්
ආයුබෝවන් මගේ නම C-3PO.
අපට සිටින රොබෝ ගණන 2
රොබෝට වැඩ ඇත
රොබෝ වැඩ අවසානයි අබලි කරන්න
R2-D2 අබලි කරමින් පවතී
තව රොබෝවරු සිටිති
C-3PO අබලි කරමින් පවතී
C-3PO අවසාන රොබෝය
අපට සිටින රොබෝ ගණන 0

එය වැඩ කරන ආයුරු
population විචල්‍ය Robo පංතියට අයත්ය (class variable).
එය Robot.population ලෙස ලබාගත හැක.

name නිර්මාණය කරණ වස්තුවේ විචල්‍යයකි. (object variable).
එය වස්තුල තුලදී self.name ලෙස ලබාගත හැක.

__init__ ක්‍රමය වස්තුවේ ආරම්භක අවස්ථාවේ යොදාගනී
එහිදී අප population පංති විචල්‍යය 1 කින් වැඩි කරමු

how_many යන ශ්‍රිතය, පංතියට අයත් ක්‍රමයකි.
discard නිර්මාණය වූ වස්තුවේ ක්‍රමයකි.
එහිදී population පංති විචල්‍යය 1 කින් අඩු කෙරේ

ගණිත උදාහරණයක්
import math

class Point:

```
    def move(self, x, y):
        self.x = x
        self.y = y
    def reset(self):
        self.move(0, 0)
    def calc_dist(self, other_point):
        return math.sqrt(
                (self.x - other_point.x)**2 +
                (self.y - other_point.y)**2)

# භාවිතය

point1 = Point()
point2 = Point()

point1.reset()
point2.move(5,0)
print(point2.calc_dist(point1))
assert (point2.calc_dist(point1) == point1.calc_dist(point2))
point1.move(3,4)
print(point1.calc_dist(point2))
print(point1.calc_dist(point1))
```

බැංකු ගිණුම් උදාහරණය

```
class BankAcc(object):
    defaultNumber = 1       # Class Attribute

    def __init__(self, name, balance = 0):
        self.name = name
        self.balance = balance
        self.accNumber = BankAcc.defaultNumber
        BankAcc.defaultNumber = BankAcc.defaultNumber + 1

    def deposit(self, amount):
        self.balance += amount

    def withdraw(self, amount):
        if self.balance < amount:
            print('Not enough balance!')
        else:
            self.balance -= amount

    def getBalance(self):
        return self.balance

if __name__ == '__main__':
    Obj = BankAcc('Indika', 1000)
    Obj.deposit(1000)
    print(Obj.getBalance())
    Obj.withdraw(500)
    print(Obj.getBalance())
```

පරිගණක ක්‍රීඩා උදාහරණයක්

```
import random

class WorldMap(object):
    def __init__(self, width, height):
        self.width = width
        self.height = height
        self.map = [[None for x in range(self.width)] for y in
range(self.height)]

    def is_occupied(self, x, y):
        ''' Checks if a given space on the map and returns True if
occupied. '''
        return self.map[x][y] is not None

world = WorldMap(100, 100)

class Entity(object):
    #def __init__(self, x, y):
        #self.set_position(x, y)
        # TODO: prompt for new x & y when (x,y) is already
occupied
        # PROPOSED: new __init__ below:

    def __init__(self, x, y):
        if world.is_occupied(x, y):
            print("oops, there's someone here already")
        else:
            self.set_position(x, y)

    def set_position(self, x, y):
        self.x = x
        self.y = y
        world.map[x][y] = self
```

```
    def remove(self):
        world.map[self.x][self.y] = None

    def distance(self, other):
        return [abs(other.x-self.x), abs(other.y-self.y)]

class Character(Entity):

    def __init__(self, x, y, hp):
        Entity.__init__(self, x, y)
        self.hp = hp
        self.items = []
        self.protection = 0

    def move(self, direction):
        '''
            Moves a character one space in a given direction. Takes
as input a
            direction 'left', 'right', 'up' or 'down'. Allows wrapping of the
            world map (eg. moving left from x = 0 moves you to x =
-1)
        '''
        x, y = 0, 0

        if direction == 'left':
            x = -1
        elif direction == 'right':
            x = 1
        elif direction == 'up':
            y = 1
        elif direction == 'down':
            y = -1
        else:
```

```
            print("Please enter a valid direction: 'left', 'right', 'up', or
'down'")
            return

        new_x, new_y = ((self.x + x) % world.width), ((self.y + y) %
world.height)
        if world.is_occupied(new_x, new_y):
            print('Position is occupied, try another move.')
        else:
            self.remove()
            self.set_position(x, y)
            if isinstance(self, Wizard):  # Provides a way for the
wizard to regen. mana
                self.mana += 1

    def attack(self, enemy):
        damage = 10
        if self.can_attack(enemy):
            if not enemy.has_protection():
                enemy.lose_health(damage)
            else:
                enemy.lose_protection(damage)

    def power_attack(self,enemy):  # power attack
        damage = 20
        if self.can_attack(enemy):
            if not enemy.has_protection():
                enemy.lose_health(damage)
            else:
                self.lose_protection(damage)

    def can_attack(self, enemy):
        enemy_dx, enemy_dy = self.distance(enemy)
        return (enemy_dx == 1 and enemy_dy == 0)
```

```
    def gain_health(self): # gain hp....
        self.hp += 10

    def lose_health(self, reduction):
        self.hp = max(self.hp - reduction, 0)

    def gain_protection(self):
        # its provide  a kind of protection from enemy attack
        self.protection += 4

    def lose_protection(self, reduction):
        self.protection = max(self.protection - reduction, 0)

    def has_protection(self):
        return self.protection > 0

class Enemy(Character):
    def __init__(self, x, y, hp):
        Character.__init__(self, x, y, hp)

    def challenge(self, other):
        print("Let's fight!")

class Wizard(Character):
    def __init__(self, x, y, hp, mana):
        Character.__init__(self, x, y, hp)
        self.mana = mana

    def cast_spell(self, enemy):
        if self.can_attack(self, enemy) and self.mana >= 5:
            self.mana -= 5
            for i in range(random.randint(1, 5)):
                self.attack(self, enemy)

    def heal(self, char):
```

```
        #if self.can_attack(self, enemy) and self.mana >= 5:
        if self.mana >= 5:
            self.mana -= 5
            char.hp += 15

class Archer(Character):
    def __init__(self, x, y, hp):
        Character.__init__(self, x, y, hp)

    def range_attack(self, enemy):
        enemy_dx, enemy_dy = self.distance(enemy)
        if enemy_dx <= 5 and enemy_dy == 0:
            enemy.hp -= 5

if __name__ == "__main__":
    print("testing: " + __file__ )
    alice = Character(10,10,100)
    bob = Character(11,10,100)
    clare = Wizard(15,10,60,50)
    dan = Archer(6,10,30)
    character_list = [alice, bob, clare, dan]
    for character in character_list:
        print(character.__dict__)
```

කැමති විෂය
කැමති වෙලාවක
කැමති තැනක
නිදහසේ ඉගන ගන්න
පාඩම් සහ ප්‍රශ්න

Shilpa64.lk

ශිල්පසයුර Google for Education CSSL COMPUTER SOCIETY OF SRI LANKA

AlgoHack කුඩා අවදියේදී ළමුනට පරිගණක විද්‍යාව සහ ක්‍රම ලේඛනය ඉගැන්වීමට ශිල්ප සයුර කල ආරම්භයකි. එයට Google for Education සහ ශ්‍රී ලංකා පරිගණක සංගමය සහාය ලබා දේ

AlgoHack : පයිතන් #4

දන්න අකෘති - ලැයිස්තු ශ්‍රිත - ගොනු - වස්තු