# Python Programming Assignment 1 Report

He Hengkang

## 1、Emirp

Search for Emirps incrementally starting from 0.
How can I determine that the current number is Emirp?First, it's prime. Second, it can't be nonpalindromic.
How do you know a number is prime?This number is not divisible by the number from 1 to itself taking the square root of +1.
How to judge a number as a palindrome number?Reverse it to a new number, and see if it's the same.

**Code:**

```
# ------------------------------------------------------- #
# abstract : assignment 1 Question 1 Emirp 回文质数
# author : hhk
# time : 2022.10.13
# ------------------------------------------------------- #
from math import sqrt

def is_prime(num):
    """
    whether it is prime or not 判断是否为质数
    """
    for rea in range(2,int(sqrt(num)+1)):
        if num==1 : return False
        if num%rea==0:
            return False
    return True

def is_palindrome(num):
    """
    whether it is Emirp or not 判断回文是不是质数、回文是否等于自己
    """
    temp=num
    total=0
    while temp>0:
        total=total * 10+temp % 10
        temp//=10
    return is_prime(total) and (total!=num)
```

```
# 输入
number = int(input(’Please input an integer : ’))
a = 1
ans = []
while(number>0):
    if is_palindrome(a) and is_prime(a) :
        ans +=[a]
        number -=1
    a +=1

# 输出
for i in range(len(ans)):
    print(str(ans[i]).rjust(6), end=’’)       # .rjust 补全 n 个字符
    if (i+1)%10 == 0:                         # 每 10 个数换行一次
        print(”\n”)
```

## 2、Integral

I set up a dict to hold available functions.Index to the corresponding function based on the user's input character.

In the function call part, we repeatedly judge the user's input, such as judging the size relationship of ab, such as judging whether n is a positive integer, so as to increase the robustness of this program.

**Code:**

```
# ------------------------------------------------------ #
# abstract : assignment 1 Question 2 Integral 数值积分
# author : hhk
# time : 2022.10.13
# ------------------------------------------------------ #
import math
from math import sin,cos,tan

# set up a dict to hold available functions
func_dict = {
    ”sin”: sin,
    ”cos”: cos,
    ”tan”: tan,
}

def Intergral(funcName,a,b,n):                # 实参下划线，形参区分大小写
```

```
    """
    computer integral 计算数值积分
    """
    func = func_dict[funcName]     # 函数名能直接传
    ans = 0
    for i in range(1,n+1):
        ans += (b-a)/n*func(a+(b-a)/n*(i-1/2))
    print("answer is ",ans,end="\n")
    return ans

# --------- main part --------- #
while(1):
    print("hello,welcome!")
    # input function name
    func_name= input("please input a trigonometric function  ")            # input 进来的都是 str
    if not func_dict.__contains__(func_name):  # not 表示取反，~按位取反
        print("input wrong function\n")
        continue
    # input a,b
    a = float(input("please input a bigger interval end points  "))
    b = float(input("please input a smaller interval end points  "))
    if b<a:
        print("input wrong a and b\n")
        continue
    # input n
    n = input("please input a number of sub-intervals n  ")
    if (not str.isdigit(n)) or int(n)<0:         # 判断字符串是否为整数
        print("input wrong n\n")
        continue
    n = int(n)
    # func_name,a,b,n = "sin",0,math.pi,20
    Intergral(func_name,a,b,n)
    print("bye~")
    break
```

## 3、Locker puzzle

Two loops, one loop for locker, loop for each student in each locker.
In the second loop, If the locker's number is divisible by the student's number, the student will change the locker once.

**Code:**

```
## ------------------------------------------------------- #
# abstract : assignment 1 Question 3 Locker puzzle
# author : hhk
# time : 2022.10.13
# ------------------------------------------------------- #
print("Locker puzzle")
lockers=100
students=100

resultLockerIsOpen={}
print("\nLockers"),
for a in range(1,lockers+1):        # loop all lockers
    resultLockerIsOpen[a] = False
    for b in range (1,students+1):    # loop all students
        if a%b == 0:
            resultLockerIsOpen[a]= not resultLockerIsOpen[a]
    # print result
    if resultLockerIsOpen[a]:
        print("{},".format(a),end=" ")
print("\nare open")
```

## 4、Tree

It creates a Node class, a Tree class. When Tree class is initialized, a root Node is generated.

The insert and search functions compare the input value to the current Node value, moving to the right if the input value is large, and to the left if the input value is small.

The print function iterates through the left side of the node, then the middle, and then the right side.

**Code:**

```
# ------------------------------------------------------- #
# abstract : assignment 1 Question 4 Tree
# author : hhk
# time : 2022.10.13
# ------------------------------------------------------- #

# --------- definition --------- #
class Node:
    def __init__(self, data=None):
        self.left = None
        self.right = None
        self.data = data

    def insert(self, data):
        if self.data:
            if data < self.data:
                if self.left is None:
                    self.left = Node(data)
                else:
                    self.left.insert(data)
            elif data > self.data:
                if self.right is None:
                    self.right = Node(data)
                else:
                    self.right.insert(data)
        else:
            self.data = data

    def Print(self):
        if self.data is None:
            return
        if self.left:
            self.left.Print()
        print(self.data,end=” ”),
        if self.right:
            self.right.Print()

    def Search(self,val):
        if self.data:
            if self.data == val:
                print(self)
                return self
            elif self.data>val:
                if self.left is None:
                    print(’left’)
                    return None
                else:
                    self.left.Search(val)
            elif self.data<val:
                if self.right is None:
                    print(’right’)
                    return None
                else:
                    self.right.Search(val)
        else:
            return None

class Tree:
    def __init__(self,data = None):
        """
        initialize root node
        """
        self.root = Node(data)           # 根节点

    def GetRoot(self):
        """
        return root node
        """
        return self.root


    def Insert(self,val):
        """
        insert node
        """
        self.root.insert(val)

    def PrintTree(self):
        """
        print the tree
        """
        if self.root.data is None:
            print("Nothing in tree\n")
        else:
            print("Values in tree")
            self.root.Print()
            print('\n')

    def SearchTree(self,val):
        """
        search value by input val
        return node if value == input, return None otherwise
        """
        print(self.root.Search(val))
        # a = self.root.Search(val)
        # print(a)
        # return a
        # return self.root.Search(val)
    def Delete(self,val=None):
        """
        delete node according to input value;
        delete node, whose value equals to input, and its children
        """
        if val is None:
            del self.root
            self.root = Node()
            return
        if self.SearchTree(val):
            a = self.SearchTree(val)
            del a
        else:
            print("The value does not exist in the tree")
            return


# --------- main part --------- #

tree = Tree()
tree.Insert(8)
tree.Insert(10)
tree.Insert(1)
tree.Insert(6)
tree.Insert(15)
tree.PrintTree()

tree.PrintTree()

content = [1,5,99,15,100,35,23,20,16,13]
for num in content:
    tree.Insert(num)
tree.PrintTree()
```

# 5、Permutation

Use the idea of backtracking to solve.Create a path variable to store the usage of the numbers in the list.

**Code:**

```
# ------------------------------------------------------ #
# abstract : assignment 1 Question 5 Permutation
```

```python
# author : hhk
# time : 2022.10.13
# ------------------------------------------------------- #
import copy

result = []  # The global variable
path = []

# --------- definition --------- #
def BackTracking(nums,used):
    """
    :param nums: The List of numbers to be arranged
    :param used: Store a Boolean list of numbers in nums that have not been used
    :return:
    """
    if(len(path) == len(nums)):
        result.append(path.copy())          # 必须要path.copy()，否则list更新错误
        return
    for i in range(len(nums)):
        if(used[i] is True):
            continue
        used[i] = True
        path.append(nums[i])
        BackTracking(nums,used)
        path.pop()
        used[i] = False

def permute(nums):
    used = []
    for i in range(len(nums)):    # define an all-false list
        used.append(False)
    BackTracking(nums,used)
    return result


# --------- main --------- #
nums = [1,2,3]
ans = permute(nums)
print(ans)
```