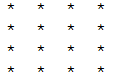
Get input from user and print ‘candy’ corresponding to that input

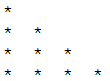
x = int(input(**"Enter the number of candies : "**))  
i = 1  
**while** i <= x:  
 print(**"Candies "**, end=**""**)  
 i += 1

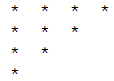
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

x = int(input(**"Enter the number of candies : "**))  
i = 1  
**for** i **in** range(x):  
 print(**"Candies "**, end=**""**)  
 i += 1

patterns

**for** i **in** range(4):  
 **for** j **in** range(4):  
 print(**" \* "**, end=**""**)  
 print()

**for** i **in** range(4):  
 **for** j **in** range(i+1):  
 print(**" \* "**, end=**""**)  
 print()

**for** i **in** range(4):  
 **for** j **in** range(4-i):  
 print(**" \* "**, end=**""**)  
 print()

Examples

x = **"Mosh"**y = **"Hamdani"**result = **f"{**x**} [{**y**}] is my favorite"**print(result)

# Mosh [Hamdani] is my favorite

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

course = **"Bonjour le monde, Je suis Basir Payenda"**print(course.find(**'le'**))

*# 8*

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

course = **"Bonjour le monde, Je suis Basir Payenda"**print(**"Bonjour" in** course)

*# True*

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**import** math  
  
print(math.ceil(2.9)) *# 3*print(math.floor(2.1)) *# 2*print(math.factorial(5)) *# 120*print(math.cos(90)) *# -0.4480736161291701*print(math.pi) *# 3.141592653589793*print(math.remainder(10, 3)) *# 1*print(math.e) *# 2.718281828459045*print(math.log(10, 10)) *# 1.0*print(math.log2(10)) *# 3.321928094887362*print(math.log10(10)) *# 1.0*print(math.sqrt(9)) *# 3*print(math.pow(9, 2)) *# 81.0*print(math.exp(2)) *# 7.3890560989306*

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Price of a house is $1M, if buyer has good credit, they need to put down 10% otherwise they need to put down 20%. Print the down payment

price = 1000000  
has\_good\_credit = **False  
  
if** has\_good\_credit:  
 down\_payment = 0.1 \* price  
**else**:  
 down\_payment = 0.2 \* price  
  
print(**f"Down Payment : {**down\_payment**}"**)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

If applicant has good credit and has no criminal record. He is eligible for loan.

has\_good\_credit = **True**has\_criminal\_record = **False  
  
if** has\_good\_credit **and not** has\_criminal\_record:  
 print(**"Eligible for loan"**)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

POUND TO KILO CONVERTER:

weight = int(input(**"Weight : "**))  
lb\_kg = input(**"lb or kg : "**)  
lb\_kg.lower()  
  
**if** lb\_kg == **"lb" or** lb\_kg == **"pound" or** lb\_kg == **"kb" or** lb\_kg == **"pounds"**:  
 result = weight \* 0.45  
 print(**f"You are {**result**} kilo grams."**)  
**else**:  
 result = weight // 0.45  
 print(**f"You are {**result**} lbs."**)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

BUILDING A GUESSING GAME

secret\_number = 9  
guess\_count = 0  
guess\_limit = 3  
  
**while** guess\_count < guess\_limit:  
 guess = int(input(**"Guess : "**))  
 guess\_count += 1  
 **if** guess == secret\_number:  
 print(**"You won!"**)  
 **break  
else**:  
 print(**"You failed!"**)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

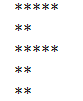
BUILDING THE CAR GAME

command = **""  
while** command != **"quit"**:  
 command = input(**"> "**).lower()  
  
 **if** command == **"help"**:  
 print(**'''  
 start - to start the car  
 stop - to stop the car  
 quit - to exit  
 '''**)  
 **elif** command == **"start"**:  
 print(**"Car started and ready to go"**)  
 **elif** command == **"stop"**:  
 print(**"Car stopped."**)  
 **elif** command == **"quit"**:  
 print(**"Game exit"**)  
 **else**:  
 print(**"Command not found!"**)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

prices = [10, 20, 30]  
sum = 0  
**for** i **in** prices:  
 sum += i  
print(sum)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

num = [5, 2, 5, 2, 2]  
**for** i **in** range(len(num)):  
 print(num[i] \* **"\*"**) **\_\_\_\_\_\_\_\_\_\_\_\_\_or\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

num = [5, 2, 5, 2, 2]  
**for** i **in** num:  
 print(i \* **"\*"**)

num = [5, 2, 5, 2, 2]  
**\_\_\_\_\_\_\_\_\_\_\_\_\_or\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  
**for** i **in** num:  
 output = **""  
 for** j **in** range(i):  
 output += **"\*"** print(output)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

FINDING THE MAXIMUM NUMBER

nums = [120, 14, 334, 129, 1000, 555]  
max = nums[0]  
**for** num **in** nums:  
 **if** num > max:  
 max = num  
print(max)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

TWO DIMENSION LIST

matrix = [  
 [1, 2, 3],  
 [4, 5, 6],  
 [7, 8, 9]  
]  
  
**for** x **in** matrix:  
 **for** y **in** x:  
 print(y)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

phone\_num = input(**"Phone : "**)  
digit\_mapping = {  
 **'1'** : **'One'**,  
 **'2'** : **'Two'**,  
 **'3'** : **'Three'**,  
 **'4'** : **'Four'**}  
output = **""  
for** x **in** phone\_num:  
 output += digit\_mapping.get(x, **"!"**) + **" "**print(output)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*# EMOJI CONVERTER*

state = input(**"How are you today ? "**)  
state\_splitted = state.split(**" "**)  
emojis = {  
 **":)"** : **"Happy"**,  
 **":("** : **"Sad"**}  
  
output = **""  
for** x **in** state\_splitted:  
 output += emojis.get(x, x) + **" "**print(output)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*# RECURSIVE FUNCTIONS*

**def** tri\_recursion(k):  
 **if**(k>0):  
 result = k+tri\_recursion(k-1)  
 print(result)  
 **else**:  
 result = 0  
 **return** result  
  
print(**"\n\nRecursion Example Results"**)  
tri\_recursion(6)

# 1 3 6 10 15 21

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**class** Person:  
 **def** \_\_init\_\_(self, name):  
 self.name = name  
 **def** talk(self):  
 print(**f"Hey this is {**self.name**}. You know me"**)  
  
person1 = Person(**"John Doe"**)  
print(person1.talk())

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*# INHERITANCE*

**class** Mammal:  
 **def** talk(self):  
 print(**"talk"**)  
**class** cat(Mammal):  
 **pass  
class** dog(Mammal):  
 **def** bark(self):  
 print(**"bark"**)

dog1 = dog()  
dog1.walk()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**def** find\_max(lists):  
 max\_num = lists[0]  
 **for** y **in** lists:  
 **if** y > max\_num:  
 max\_num = y *# not y = max\_num*  
 print(max\_num)  
  
find\_max([12, 10014, 15, 150, 17, 18])

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**import** random  
  
print(random.random()) *# 0.01752739145883464*print(random.randint(10, 20)) *# random num between 10 to 20*members = [**'John'**, **'Mary'**, **'Bob'**, **'Mosh'**]  
leader = random.choice(members)  
print(leader)

*# Choose members by random*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**import** random  
  
**class** Dice:  
 **def** roll(self):  
 first = random.randint(1, 6)  
 second = random.randint(1, 6)  
 **return** first, second *# python will read this as tuple*

person1 = Dice()  
print(person1.roll())

\_\_\_\_\_\_\_\_\_\_\_\_\_\_(path)\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**from** pathlib **import** Path  
  
path = Path(**"emails"**)  
print(path.exists()) *# Checks if emails directory exists*print(path.mkdir()) *# Make emails directory*print(path.rmdir()) *# Remove emails directory*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**from** pathlib **import** Path  
  
path = Path()  
**for** file **in** path.glob(**'\*.py'**):  
 print(file) # Finds all files with *'*.py*'* extension

**for** file **in** path.glob(**'\*.xlsx'**):

print(file) # Finds all files with *'*.xlsx*'* extension

**for** file **in** path.glob(**'\*'**):

print(file) # Finds all files and directories in current path

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

str = **"Hello World"**intro = **"This is Basir Payenda"**print(**f"{**str**}, {**intro**}. I am 23 years old"**)

# The code above is identical to one below here.

print(**"{}, {}. I am 23 years old"**.format(str, intro))

# Hello World, this is Basir Payenda. I am 23 years old

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

intro = **"hello world"**print(help(str.lower))  
*# Help on built-in function lower:  
# lower() method of builtins.str instance  
# Return a copy of the string converted to lowercase.  
# None*print(intro.title())  
*# Hello World*

print(type(intro))

*# <class 'str'>*print(intro.count(**"l"**))  
*# 3*print(intro.find(**"w"**))  
*# 6*print(dir(str)) *# lists all methods of string  
# ['\_\_add\_\_', '\_\_class\_\_', '\_\_contains\_\_', '\_\_delattr\_\_', '\_\_dir\_\_', '\_\_doc\_\_', '\_\_eq\_\_',  
# '\_\_format\_\_', '\_\_ge\_\_', '\_\_getattribute\_\_', '\_\_getitem\_\_', '\_\_getnewargs\_\_', '\_\_gt\_\_' ... ]*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

print(abs(-3))  
print(round(2.78))  
print(round(2.78,1)) *# round to the 1st digit after decimal*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

course = [**'History'**, **'Math'**, **'Physics'**]  
course\_2 = [**'Art'**, **'Education'**]  
  
*# APPEND*course.append(course\_2)  
print(course)  
*# ['History', 'Math', 'Physics', ['Art', 'Education']]  
  
# INSERT*course.insert(0, course\_2)  
print(course)  
*# [['Art', 'Education'], 'History', 'Math', 'Physics']  
  
# EXTEND*course.extend(course\_2)  
print(course)  
*# ['History', 'Math', 'Physics', 'Art', 'Education']*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*# Intersection, difference, union in* ***SETs***

course = {**"Python"**, **"Database"**, **"Network"**, **"Data Structure"**}  
art\_course = {**"Python"**, **"Art"**, **"Network"**, **"Data Structure"**, **"Design"**}  
  
print(course.intersection(art\_course))  
*# Intersection : Common courses  
# {'Python', 'Data Structure', 'Network'}*print(art\_course.difference(course))  
*# {'Art', 'Design'}*print(course.union(art\_course))  
*# {'Network', 'Database', 'Art', 'Design', 'Python', 'Data Structure'}*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*# Empty list*emp\_list = []  
emp\_list = list()

*# Empty Tuples*emp\_tuple = ()  
emp\_tuple = tuple()  
  
*# Empty Set*emp\_set = {} *# This isn't right, it sets an empty dictionary*emp\_set = set()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

person = {  
 **"name"** : **"Basir Payenda"**,  
 **"age"** : **"23"**,  
 **"email"** : **"basir.payenda@gmail.com"**,  
 **"address"** : **"Quebec Montreal"**}

*# Use update(), if you want to update multiple keys*  
person.update({**"email"** : **"mr.basir@gmail.com"**, **"address"** : **"Canada/Vancouver/Google"**})  
print(person)  
*# {'name': 'Basir Payenda', 'age': '23', 'email': 'mr.basir@gmail.com', 'address': 'Canada/Vancouver/Google'}*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*# \*args and \*\*kwargs allows us to accept an arbitrary numbers of  
# positional or keyword arguments and they necessarily don't have # to be args and kwargs . Args and kwargs are conventions, but you # can change it  
# \*\* means you can pass keyword arguments as well*

**def** student\_info(\*args, \*\*kwargs):  
 print(args)  
 print(kwargs)  
  
student\_info(**'Math'**, **'Art'**, name=**"John"**, age=22)  
*# ('Math', 'Art')  
# {'name': 'John', 'age': 22}*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**def** student\_info(\*args, \*\*kwargs):  
 print(args)  
 print(kwargs)  
  
  
courses = (**'Math'**, **'Art'**)  
info = {**'name'**: **'John'**, **'age'**: 22}  
  
student\_info(courses, info)  
*# (('Math', 'Art'), {'name': 'John', 'age': 22})  
# {}*

*# If you see \* or \*\* while calling arguments*student\_info(\*courses, \*\*info)  
*# ('Math', 'Art')  
# {'name': 'John', 'age': 22}*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**def** find\_index(to\_search, index):  
 **for** i, value **in enumerate**(to\_search):  
 **if** value == index:  
 **return** i  
  
 **return** -1  
  
  
courses = [**"Java"**, **"C++"**, **"JS"**, **"Python"**]  
  
res = find\_index(courses, **"Python"**)  
print(res)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*# How to search for pip help and help for intalling packages ?*>>> pip help  
>>> pip help install  
  
*# Search, install and uninstall 'pympler' package*>>> pip search pympler  
>>> pip install pympler  
>>> pip list  
>>> pip uninstall pympler

*# How to know if you are using the latest version of a package?*>>> pip list --outdated  
>>> pip list -o  
  
*# How to install the latest version of a package. e.g. setuptools?*>>> pip install -U setuptools *# U for upgrade or*

>>> pip install -upgrade setuptools

*# How to provide someone with list of packages you have installed?*>>> pip freeze  
>>> pip freeze > requirement.txt  
>>> pip freeze --local > requirement.txt  
  
*# How do that someone can install from our exported .txt file?*>>> pip install -r requirement.txt *# r for requirement file*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

def myfunc1():  
 x = "John"  
  
 def myfunc2():  
 nonlocal x  
 x = "Hello"print(x)  
  
 myfunc2()  
 print(x)  
  
myfunc1()

*# Hello*

*# Hello*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

names = ['Basir', 'Bashir', 'Khan', 'Asif', 'Rahman']  
  
if 'Nasir' not in names:  
 print("Nasir is not in the list!")

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

def mx(x, y):  
 if x > y:  
 return x  
 else:  
 return y  
  
  
*# You can use function above or short-form lambda function*mx = lambda x, y: x if x > y else y  
  
print(mx(5,10))

*# 10*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**def** sqr(lst):  
 lst2 = []  
 **for** i **in** lst:  
 lst2.append(i\*\*2)  
 **return** lst2  
  
  
*# Or you can simply use lambda function here*n = [12, 13, 15, 16, 20, 30]  
print(sqr(n))  
  
print(list(map(**lambda** a: a \*\*2 , n)))

*# [144, 169, 225, 256, 400, 900]*

*# [144, 169, 225, 256, 400, 900]*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*# If a function doesn’t return a value, Python will add ‘return None’ by default to that function.*

>>> spam = print(‘Hello World’)

Hello World

>>> spam == None

True

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*# You can cause the program to terminate or exit by calling the* **sys.exit()** *function.*

import sys

while True:

print('Type exit to exit.')

response = input()

if response == 'exit':

sys.exit()

print('You typed', response, '.')

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

False or False # False

False and False # False

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

print('Hello world!', 'How\'re you doing', sep=',', end=' ')

# Hello world!,How're you doing

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

def spam():

print(eggs)

eggs = 100

spam()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# the **global** statement

def spam():

global eggs

eggs = 1000

eggs = 10

spam()

print(eggs)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

cats = []

while True:

print('What is the name of cat #' + str(len(cats) + 1 ) + ' ,or type \'exit\' to exit the program!' )

output = input()

if output == 'exit':

break

cats = cats + [output] # list concatenation

for cat in cats:

print(cat)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# strings are immutable, but you can use this trick to modify it a bit

>>> txt = 'My name is Basir'

>>> new\_txt = txt[:11] + 'John Doe.'

>>> new\_txt

'My name is John Doe.'

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

>>> type(('One Item'))

<class 'str'>

>>> type(('One Item',)) # tuples with one item should end via trailling comma

<class 'tuple'>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Python uses references whenever variables must store values of mutable data types, such as lists or dictionaries. For values of immutable data types such as strings, integers, or tuples, Python variables will store the value itself.

list1 = [1, 2, 3, 4, 5]

list2 = list1

list1[2] = 'Johnny'

print(list2)

print(list1)

# another example

print("\*" \* 30)

spam = 42

eggs = spam

spam = 100

print('spam =', spam)

print('eggs =', eggs)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

>>> 'hello'.isalpha()

True

>>> 'hello123'.isalpha()

False

>>> 'hello123'.isalnum()

True

>>> 'hello'.isalnum()

True

>>> '123'.isdecimal()

True

>>> ' '.isspace()

True

>>> 'This Is Title Case'.istitle()

True

>>> 'This Is Title Case 123'.istitle()

True

>>> 'This Is not Title Case'.istitle()

False

>>> 'This Is NOT Title Case Either'.istitle()

False

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

txt = """

Hello Dear,

What is going on?

How are you guys doing? It has been long time since when didn't see each other!

"""

res = txt.split('\n')

print(res)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

>>> spam = 'SpamSpamBaconSpamEggsSpamSpam'

# Order doesn’t matter, it’s identical to spam.strip('Spam')

>>> spam.strip('ampS')

'BaconSpamEggs'

v

# Adding bullet points to copied text

import pyperclip

text = pyperclip.paste()

lines = text.split('\n')

for i in range(len(lines)):

lines[i] = '\* ' + lines[i]

result = '\n'.join(lines)

pyperclip.copy(result)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

res = '-'.join('There can be only one.'.split())

print(res)

# There-can-be-only-one.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

table\_data = [['apples', 'Nooruddin', 'cherries', 'banana'],

['Alice', 'Bob', 'Carol', 'David'],

['dogs', 'cats', 'moose', 'goose']]

def align\_printed\_list(table):

col\_width = [0] \* len(table)

for i in range(len(table)):

for j in range(len(table[0])):

if len(table[i][j]) > col\_width[i]:

col\_width[i] = len(table[i][j])

for i in range(len(table[0])):

for j in range(len(table)):

print(table[j][i].rjust(col\_width[j]), end=' ')

print()

align\_printed\_list(table\_data)

# apples Alice dogs

# Nooruddin Bob cats

# cherries Carol moose

# banana David goose