In [ ]:
```
import numpy as np
```

In [31]:
```
#task1

def solution1(mas):
    return [x*4  for x in mas  ]

def solution2(mas):
    return [x*(i+1) for (i,x) in enumerate(mas)]

def solution3(mas):
    return [i  for i in range(16) if (i%5==0 or i%3==0) ]

def solution4(mas):
    return [el  for m in mas for el in m]

def solution5(n):
    return [(i,j,k) for i in range(1,n+1) for j in range(1,n+1) for

def solution6(mas):
    return [ list(i+j for j in mas[1]) for i  in mas[0]]

def solution7(mas):
    return [ list(m[j]   for m in mas) for (j,_)  in enumerate(mas[

def solution8(mas):
    return [list(int(x) for x in m.split()) for m in mas]

def solution9(mas):
    return {chr(ord('a')+x):x**2  for x in mas}

def solution10(mas):
    return { x[0].upper() + x[1:].lower() for x in mas if len(x)>3}

solutions = {
    'solution1': solution1,
    'solution2': solution2,
    'solution3': solution3,
    'solution4': solution4,
    'solution5': solution5,
    'solution6': solution6,
    'solution7': solution7,
    'solution8': solution8,
    'solution9': solution9,
    'solution10': solution10,
}

solutions['solution1']('python')
solutions['solution5'](15)
```

Out[31]:
```
[(3, 4, 5), (5, 12, 13), (6, 8, 10), (9, 12, 15)]
```

In [170]:
```
#taskB
import re
from functools import reduce
import operator

def solution1(mas):
    #return list(map(lambda x: reduce(lambda a,b: a[-1::-1]+b[-1::-
    return list(map( lambda x:  int(re.sub(r'[,-.]','',x).strip()[-

def solution2(mas):
    return list(map(lambda x: x[0]*x[1], mas))

def solution3(mas):
    return list(filter(lambda x: (x%6==0 or x%6==2 or x%6==5),mas))

def solution4(mas):
    return list(filter(lambda x: bool(x),mas))

def solution5(mas):
    return list(map(lambda x: operator.setitem(x,'square',x['width'


def solution6(mas):
    return list(map(lambda x: dict(x,square=x['width']*x['length']))

def solution7(mas):
    return set(reduce(lambda x,y: x.intersection(y),mas))

def solution8(mas):
    return dict(reduce(lambda x,y: (operator.setitem(x, y ,x[y]+1)

def solution9(mas):
    return list(map(lambda x: x['name'], list(filter(lambda x: x['g

def solution10(mas):
    return list(filter(lambda x: reduce(lambda x,y: int(x)+int(y),


solutions = {
    'solution1': solution1,
    'solution2': solution2,
    'solution3': solution3,
    'solution4': solution4,
    'solution5': solution5,
    'solution6': solution6,
    'solution7': solution7,
    'solution8': solution8,
    'solution9': solution9,
    'solution10': solution10,
}

solutions['solution1'](['12', '25.6', '84,02', '  69-91'])
```

Out[170]: [21, 652, 2048, 1096]

Out[170]: [21, 652, 2048, 1990]

In [300]:
```
#taskC

from time import sleep
import functools
import signal


class TimeoutException(RuntimeError):
    def __init__(self, message=None):
        super().__init__(message)


def signal_handler(signum, frame):
    #print("Hello from handler")
    raise TimeoutException("Timed out")

def timeout(seconds=None):
    def decorator(func):
        if (seconds is None or seconds < 0):
            return func
        @functools.wraps(func)
        def wrapper(*args, **kwargs):
            if (seconds is not None and  seconds > 0):
                signal.signal(signal.SIGALRM, signal_handler)
                signal.setitimer(signal.ITIMER_REAL, seconds)
            result=0
            try:
                result = func(*args, **kwargs)
            finally:
                signal.setitimer(signal.ITIMER_REAL, 0)
                signal.signal(signal.SIGALRM, signal.SIG_DFL)
            return result
        return wrapper
    return decorator



@timeout(seconds=0.5)
def func():
    sleep(0.6)

try:
    func()
except TimeoutException as e:
    print(e)
```

```
Timed out
```

In [258]:
```
#taskD
def counter(func):
    def reset():
        wrapper.rdepth = 0
        wrapper.ncalls = 0

    @functools.wraps(func)
    def wrapper(*args, **kwargs):
        nonlocal tek_depth

        if tek_depth == 0:
            reset()

        tek_depth += 1
        wrapper.ncalls += 1
        wrapper.rdepth = max(wrapper.rdepth, tek_depth)

        try:
            return func(*args, **kwargs)
        finally:
            tek_depth -= 1

    tek_depth = 0
    reset()
    return wrapper

@counter
def func2(n, steps):
    if steps == 0:
        return
    func2(n + 1, steps - 1)
    func2(n - 1, steps - 1)

func2(0,5)
print(func2.ncalls, func2.rdepth)
func2(0,3)
print(func2.ncalls, func2.rdepth)
```

```
63 6
15 4
```

In [259]:
```
#does not work
def counter(func):
    @functools.wraps(func)
    def wrapper(*args, **kwargs):
        if not hasattr(wrapper,'cnt'):
            setattr(wrapper, 'cnt', 0)
        result=func(*args, **kwargs)
        cnt+=1
        return result
    return wrapper
```

```
@counter
def func2(n, steps):
    if steps == 0:
        return
    func2(n + 1, steps - 1)
    func2(n - 1, steps - 1)

func2(0,5)
func2(0,3)
print(func2.ncalls, func2.rdepth)
```

```
---------------------------------------------------------------------
---------
UnboundLocalError                         Traceback (most recent c
all last)
<ipython-input-259-d011a6825f6d> in <module>
     17     func2(n - 1, steps - 1)
     18
---> 19 func2(0,5)
     20 func2(0,3)
     21 print(func2.ncalls, func2.rdepth)

<ipython-input-259-d011a6825f6d> in wrapper(*args, **kwargs)
      5         if not hasattr(wrapper,'cnt'):
      6             setattr(wrapper, 'cnt', 0)
----> 7         result=func(*args, **kwargs)
      8         cnt+=1
      9         return result

<ipython-input-259-d011a6825f6d> in func2(n, steps)
     14     if steps == 0:
     15         return
---> 16     func2(n + 1, steps - 1)
     17     func2(n - 1, steps - 1)
     18

<ipython-input-259-d011a6825f6d> in wrapper(*args, **kwargs)
      5         if not hasattr(wrapper,'cnt'):
      6             setattr(wrapper, 'cnt', 0)
----> 7         result=func(*args, **kwargs)
      8         cnt+=1
      9         return result

<ipython-input-259-d011a6825f6d> in func2(n, steps)
     14     if steps == 0:
     15         return
---> 16     func2(n + 1, steps - 1)
     17     func2(n - 1, steps - 1)
     18

<ipython-input-259-d011a6825f6d> in wrapper(*args, **kwargs)
      5         if not hasattr(wrapper,'cnt'):
```

```
      6             setattr(wrapper, 'cnt', 0)
----> 7         result=func(*args, **kwargs)
      8         cnt+=1
      9         return result

<ipython-input-259-d011a6825f6d> in func2(n, steps)
     14     if steps == 0:
     15         return
---> 16     func2(n + 1, steps - 1)
     17     func2(n - 1, steps - 1)
     18

<ipython-input-259-d011a6825f6d> in wrapper(*args, **kwargs)
      5         if not hasattr(wrapper,'cnt'):
      6             setattr(wrapper, 'cnt', 0)
----> 7         result=func(*args, **kwargs)
      8         cnt+=1
      9         return result

<ipython-input-259-d011a6825f6d> in func2(n, steps)
     14     if steps == 0:
     15         return
---> 16     func2(n + 1, steps - 1)
     17     func2(n - 1, steps - 1)
     18

<ipython-input-259-d011a6825f6d> in wrapper(*args, **kwargs)
      5         if not hasattr(wrapper,'cnt'):
      6             setattr(wrapper, 'cnt', 0)
----> 7         result=func(*args, **kwargs)
      8         cnt+=1
      9         return result

<ipython-input-259-d011a6825f6d> in func2(n, steps)
     14     if steps == 0:
     15         return
---> 16     func2(n + 1, steps - 1)
     17     func2(n - 1, steps - 1)
     18

<ipython-input-259-d011a6825f6d> in wrapper(*args, **kwargs)
      6             setattr(wrapper, 'cnt', 0)
      7         result=func(*args, **kwargs)
----> 8         cnt+=1
      9         return result
     10     return wrapper

UnboundLocalError: local variable 'cnt' referenced before assignme
nt
```

In [30]:
```
#taskE
def chain_loop(args):
    iter_list=[iter(i) for i in args ]
    ind=0
    while(len(iter_list)>0):
        if (ind >= len(iter_list)):
            ind = 0
        try:
            x=next(iter_list[ind])
            yield x
            ind+=1
        except StopIteration:
            iter_list.pop(ind)

from itertools import tee

a = (i for i in range(3))

print(list(chain_loop(tee(a, 5))))
```

```
[0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
```

In [260]:
```
#taskF
def expand_seq(n,openb, closeb, s):
    if openb+closeb<2*n:
        if openb < n :
            Str = s + "("
            yield from expand_seq(n,openb+1, closeb,Str)
        if closeb < openb:
            Str = s + ")"
            yield from expand_seq(n,openb, closeb+1,Str)
    else:
        #print(s)
        yield s



def brackets(n):
    s= ""
    yield from expand_seq(n,0,0,s)


if __name__ == "__main__":
    n = int(input())
    for i in brackets(n):
        print(i)
```

```
3
((()))
(()())
(())()
()(())
()()()
```

In [299]:
```
#taskH

class Node:
    def __init__(self, tek_letter='!'):
        self.mas_of_children=[0 for i in range(4)]
        self.my_int=ord(tek_letter)

class Tree:
    def __init__(self, words):
        self.root=Node()
        for w in words:
            curr=self.root
            for c in w:
                if (curr.mas_of_children[ord(c)-ord('a')] == 0):
                    curr.mas_of_children[ord(c)-ord('a')]=Node(c)
                curr=curr.mas_of_children[ord(c)-ord('a')]


def printtree(koren, offset):
```

```
    if (koren==0):
        print('*')
        return



    printtree(koren.mas_of_children[3],offset+1)
    for i in range(offset):
        print("-----;",end='');

    printtree(koren.mas_of_children[2],offset+1)
    for i in range(offset):
        print("-----;",end='');

    print(chr(koren.my_int))

    printtree(koren.mas_of_children[1],offset+1)
    for i in range(offset):
        print("-----;",end='');

    printtree(koren.mas_of_children[0],offset+1)
    for i in range(offset):
        print("-----;",end='');



tree1=Tree(['adc','aab'])
print(tree1.root.my_int)
print(tree1.root.mas_of_children)
print(tree1.root.mas_of_children[0].mas_of_children)
print(tree1.root.mas_of_children[0].mas_of_children[0].mas_of_child
print(tree1.root.mas_of_children[0].mas_of_children[3].mas_of_child
printtree(tree1.root, 0)
```

```
33
[<__main__.Node object at 0x7fd0eeb7c0b8>, 0, 0, 0]
[<__main__.Node object at 0x7fd0eeb7c198>, 0, 0, <__main__.Node ob
ject at 0x7fd0eeb7c080>]
[0, <__main__.Node object at 0x7fd0eeb7c1d0>, 0, 0]
[0, 0, <__main__.Node object at 0x7fd0eeb7c160>, 0]
*
*
!
*
*
-----;-----;*
-----;-----;-----;*
-----;-----;-----;c
*
-----;-----;-----;*
-----;-----;-----;-----;-----;d
*
-----;-----;*
-----;-----;-----;*
```

```
-----;a
*
-----;*
-----;-----;*
-----;-----;a
*
-----;-----;-----;*
-----;-----;-----;b
*
-----;-----;-----;*
-----;-----;-----;-----;-----;*
-----;-----;-----;
```

In [ ]: