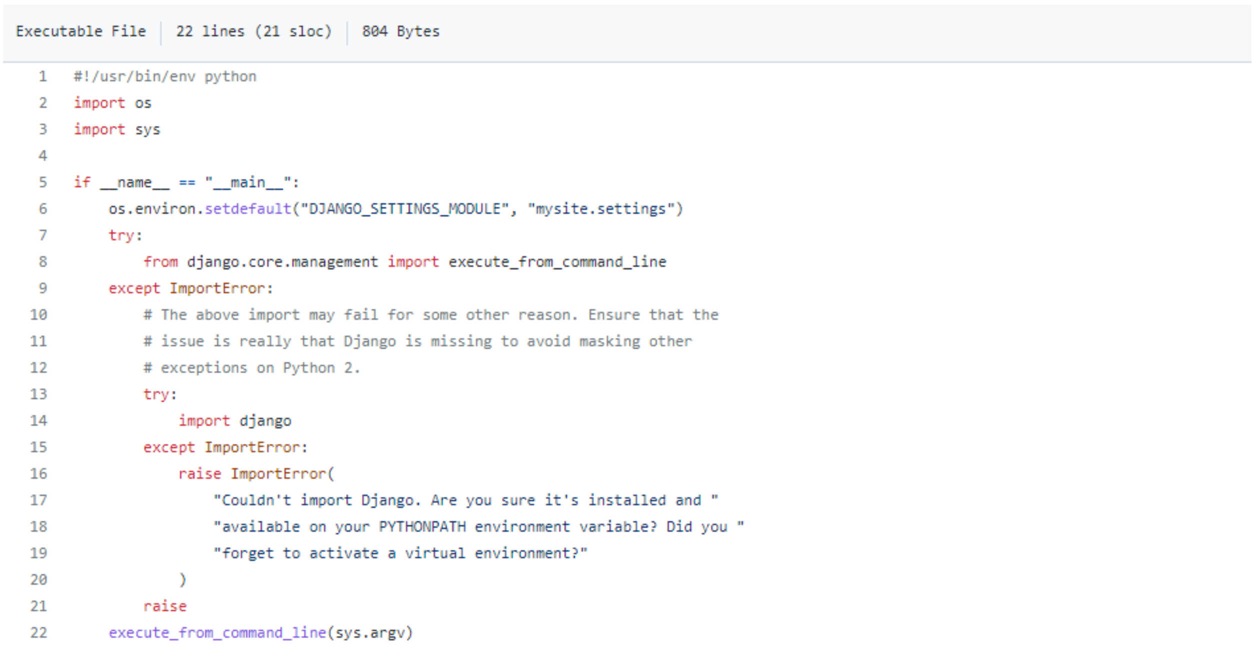
Lab-5

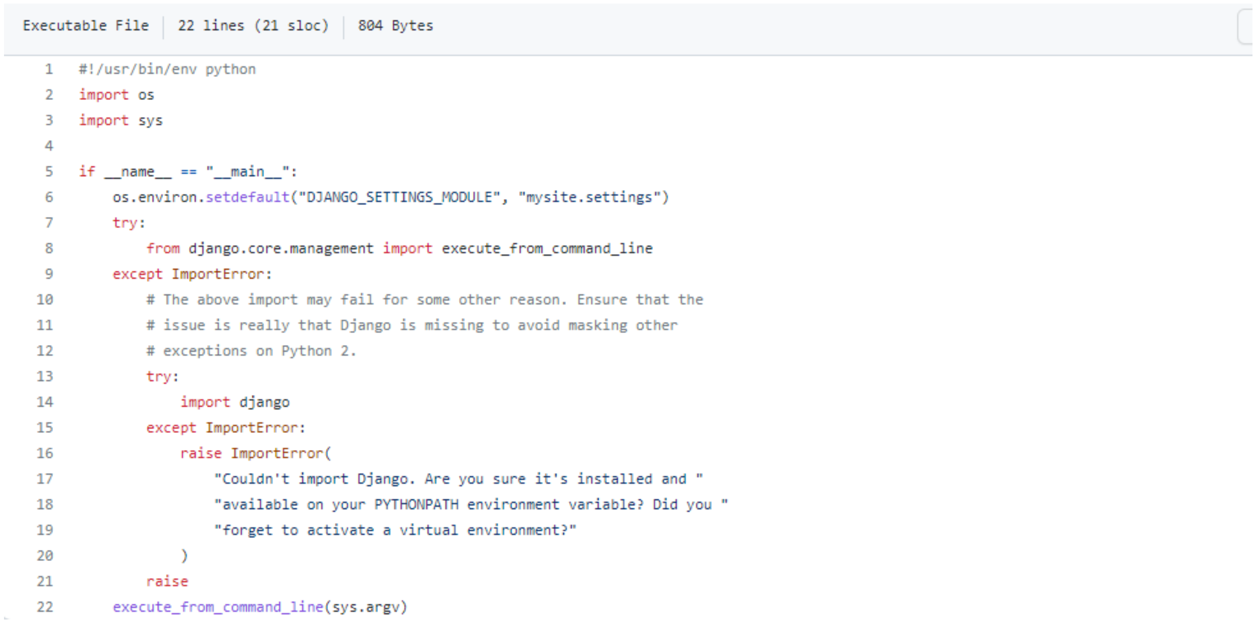
Name -Patel Ayush M.
Id -202001126

Tool - mypy



Executable File | 22 lines (21 sloc) | 804 Bytes

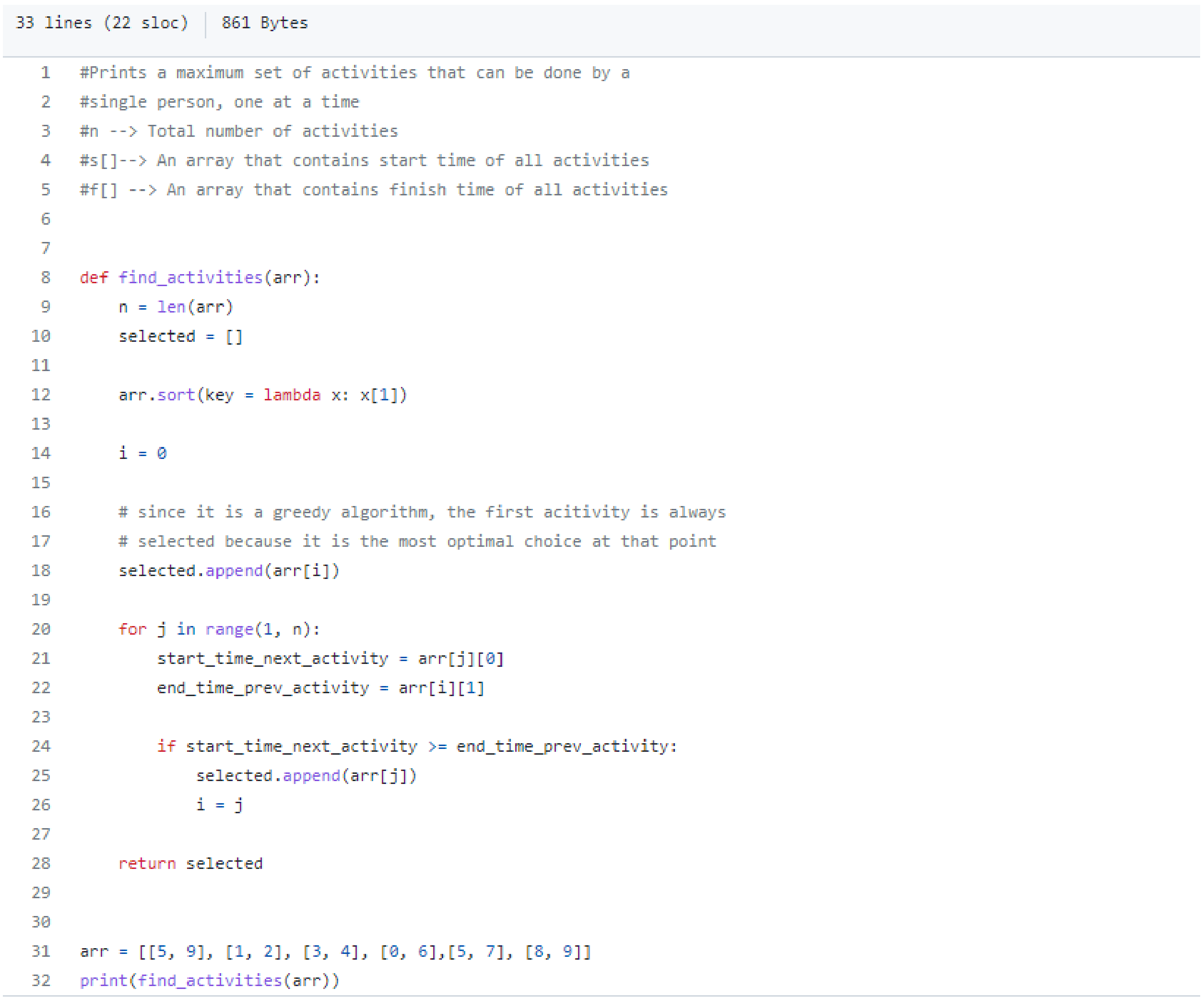
```python
#!/usr/bin/env python
import os
import sys

if __name__ == "__main__":
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "mysite.settings")
    try:
        from django.core.management import execute_from_command_line
    except ImportError:
        # The above import may fail for some other reason. Ensure that the
        # issue is really that Django is missing to avoid masking other
        # exceptions on Python 2.
        try:
            import django
        except ImportError:
            raise ImportError(
                "Couldn't import Django. Are you sure it's installed and "
                "available on your PYTHONPATH environment variable? Did you "
                "forget to activate a virtual environment?"
            )
        raise
    execute_from_command_line(sys.argv)
```


Executable File | 22 lines (21 sloc) | 804 Bytes

```python
#!/usr/bin/env python
import os
import sys

if __name__ == "__main__":
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "mysite.settings")
    try:
        from django.core.management import execute_from_command_line
    except ImportError:
        # The above import may fail for some other reason. Ensure that the
        # issue is really that Django is missing to avoid masking other
        # exceptions on Python 2.
        try:
            import django
        except ImportError:
            raise ImportError(
                "Couldn't import Django. Are you sure it's installed and "
                "available on your PYTHONPATH environment variable? Did you "
                "forget to activate a virtual environment?"
            )
        raise
    execute_from_command_line(sys.argv)
```

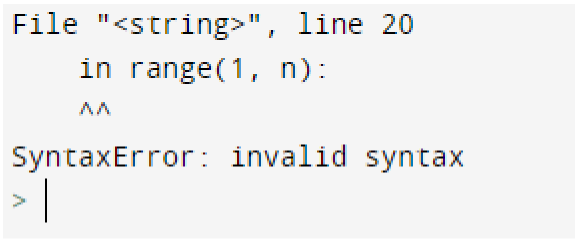
```python
import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/1.10/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'i%06y2q&4l-!nv*8oolv470b!o)!xg*^9f7^d=q10#b$wd%c_e'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    'mysite.core',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

ERROR



```
33 lines (22 sloc) | 861 Bytes
```

```
1   #Prints a maximum set of activities that can be done by a
2   #single person, one at a time
3   #n --> Total number of activities
4   #s[]--> An array that contains start time of all activities
5   #f[] --> An array that contains finish time of all activities
6
7
8   def find_activities(arr):
9       n = len(arr)
10      selected = []
11
12      arr.sort(key = lambda x: x[1])
13
14      i = 0
15
16      # since it is a greedy algorithm, the first acitivity is always
17      # selected because it is the most optimal choice at that point
18      selected.append(arr[i])
19
20      for j in range(1, n):
21          start_time_next_activity = arr[j][0]
22          end_time_prev_activity = arr[i][1]
23
24          if start_time_next_activity >= end_time_prev_activity:
25              selected.append(arr[j])
26              i = j
27
28      return selected
29
30
31  arr = [[5, 9], [1, 2], [3, 4], [0, 6],[5, 7], [8, 9]]
32  print(find_activities(arr))
```



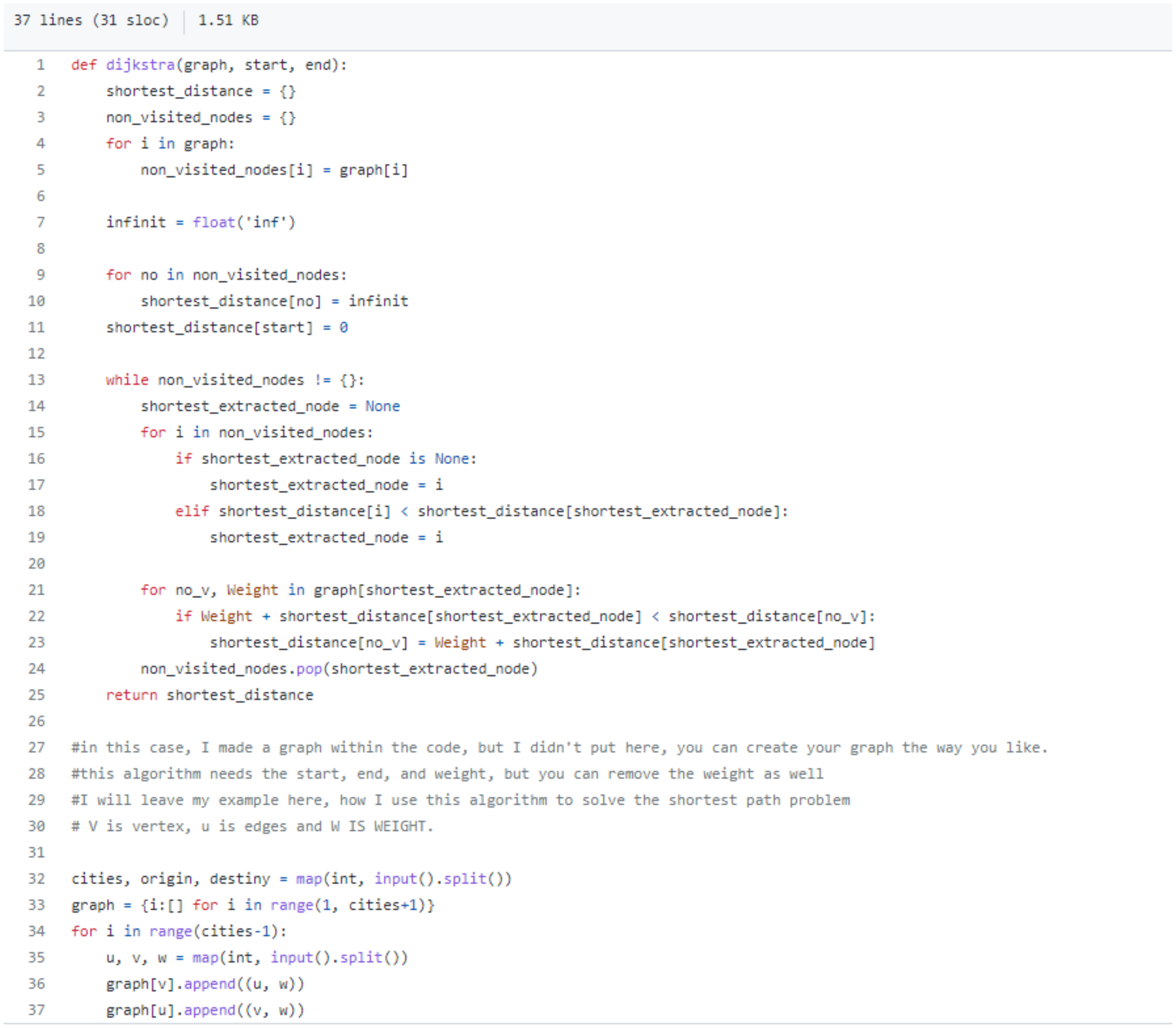
```
File "<string>", line 20
    in range(1, n):
    ^^
SyntaxError: invalid syntax
>
```

```
45 lines (33 sloc) | 980 Bytes
```

```
1  """
2  Find the min cost of tiles to cover a floor.
3  Floor is represented by 2D array where -
4  * = tile already placed
5  . = no tile
6
7  tiles available are 1*1 and 1*2 and their costs
8  are A and B
9
10 Source - https://www.geeksforgeeks.org/minimize-cost-to-cover-floor-using-tiles-of-dimensions-11-and-12/
11 """
12
13 def cost(arr, A, B):
14     n = len(arr)
15     m = len(arr[0])
16
17     ans = 0
18
19     for i in range(n):
20         j = 0
21
22         while j < m:
23             if arr[i][j] == '*':  # tile is already there
24                 j += 1
25                 continue
26
27             if j == m - 1:  # if j is pointing to last tile, you can use only 1*1 tile
28                 ans += A
29             else:
30                 if arr[i][j+1] == '.':
31                     ans += min(2 * A, B)
32                     j += 1
33                 else:
34                     ans += A
```

ERROR OUTPUT

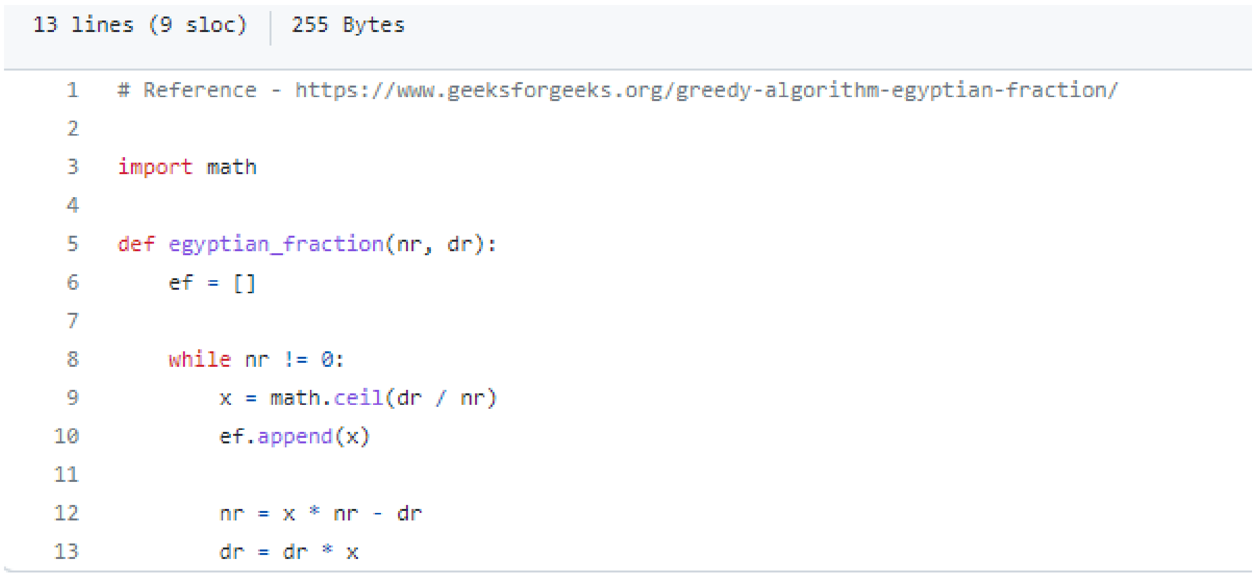
```
File "<string>", line 11
    """
Find the min cost of tiles to cover a floor.
Floor is represented by 2D array where -
* = tile already placed
. = no tile

tiles available are 1*1 and 1*2 and their costs
are A and B

Source - https://www.geeksforgeeks.org/minimize-cost-to-cover-floor-using-tiles-of-dimensions-11-and-12/
"""for i in range(n):
       ^^^
SyntaxError: invalid syntax
```



37 lines (31 sloc) | 1.51 KB

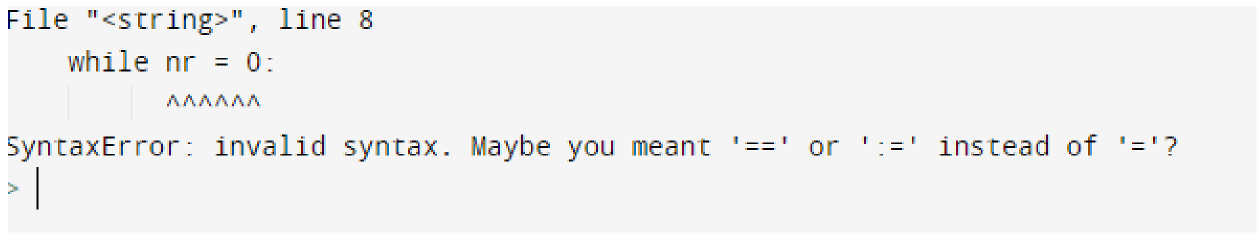
```python
def dijkstra(graph, start, end):
    shortest_distance = {}
    non_visited_nodes = {}
    for i in graph:
        non_visited_nodes[i] = graph[i]

    infinit = float('inf')

    for no in non_visited_nodes:
        shortest_distance[no] = infinit
    shortest_distance[start] = 0

    while non_visited_nodes != {}:
        shortest_extracted_node = None
        for i in non_visited_nodes:
            if shortest_extracted_node is None:
                shortest_extracted_node = i
            elif shortest_distance[i] < shortest_distance[shortest_extracted_node]:
                shortest_extracted_node = i

        for no_v, Weight in graph[shortest_extracted_node]:
            if Weight + shortest_distance[shortest_extracted_node] < shortest_distance[no_v]:
                shortest_distance[no_v] = Weight + shortest_distance[shortest_extracted_node]
        non_visited_nodes.pop(shortest_extracted_node)
    return shortest_distance

#in this case, I made a graph within the code, but I didn't put here, you can create your graph the way you like.
#this algorithm needs the start, end, and weight, but you can remove the weight as well
#I will leave my example here, how I use this algorithm to solve the shortest path problem
# V is vertex, u is edges and W IS WEIGHT.

cities, origin, destiny = map(int, input().split())
graph = {i:[] for i in range(1, cities+1)}
for i in range(cities-1):
    u, v, w = map(int, input().split())
    graph[v].append((u, w))
    graph[u].append((v, w))
```

OUTPUT -



```
13 lines (9 sloc)  |  255 Bytes

 1  # Reference - https://www.geeksforgeeks.org/greedy-algorithm-egyptian-fraction/
 2
 3  import math
 4
 5  def egyptian_fraction(nr, dr):
 6      ef = []
 7
 8      while nr != 0:
 9          x = math.ceil(dr / nr)
10          ef.append(x)
11
12          nr = x * nr - dr
13          dr = dr * x
```

ou



```
File "<string>", line 8
    while nr = 0:
          ^^^^^^
SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of '='?
> 
```