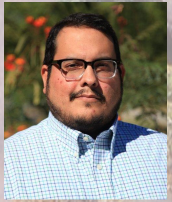
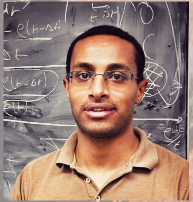
Mr.Roger’s - 20 Min Neighborhoods

Members: Kevin C, Carlos I, Michelle I
Mentors: Edgar G, Pat S, Adonay S

Theme: Mr.Roger’s Neighborhood
Theme song: Them Changes by ThunderCat





The Issue

- Dallas is dubbed “The city that hates pedestrians”
- How can we make our cities accessible?
- Give the power to the citizens by allowing them to evaluate their neighborhoods

Our Solution

Can halve household transport costs
Walking infrastructure delivers $13 benefit for every $1 spent
Improved health and wellbeing
Enhances sense of community and social cohesion
Walking infrastructure can provide a higher return than rail or road
Alleviates pressure on Melbourne's transport
Support health, infrastructure, and environmental savings to Victorian economy
Helps reduce pollution and CO₂ emissions
Increases retail trading by up to 40%

- An evaluation tool using Python by:
  - Gathering live current data
  - Computing the list of location that you can get to in 20 min
  - Allowing the user to select more than one type of transportation
  - Removing all extra and outlying data and variables
  - Using a Jupyter Notebook in Cloudy Cluster and HPC to help speed up process

Collaboration of Tasks

Kevin Chen:

- Technical Implementation/Doc
- Tech Stack Setup/Development
  - replit, git, and migration to Cloudycluster

Carlos Iglesias:

- Writing of code and pseudocode
- Presentation brainstorm and Demo recording

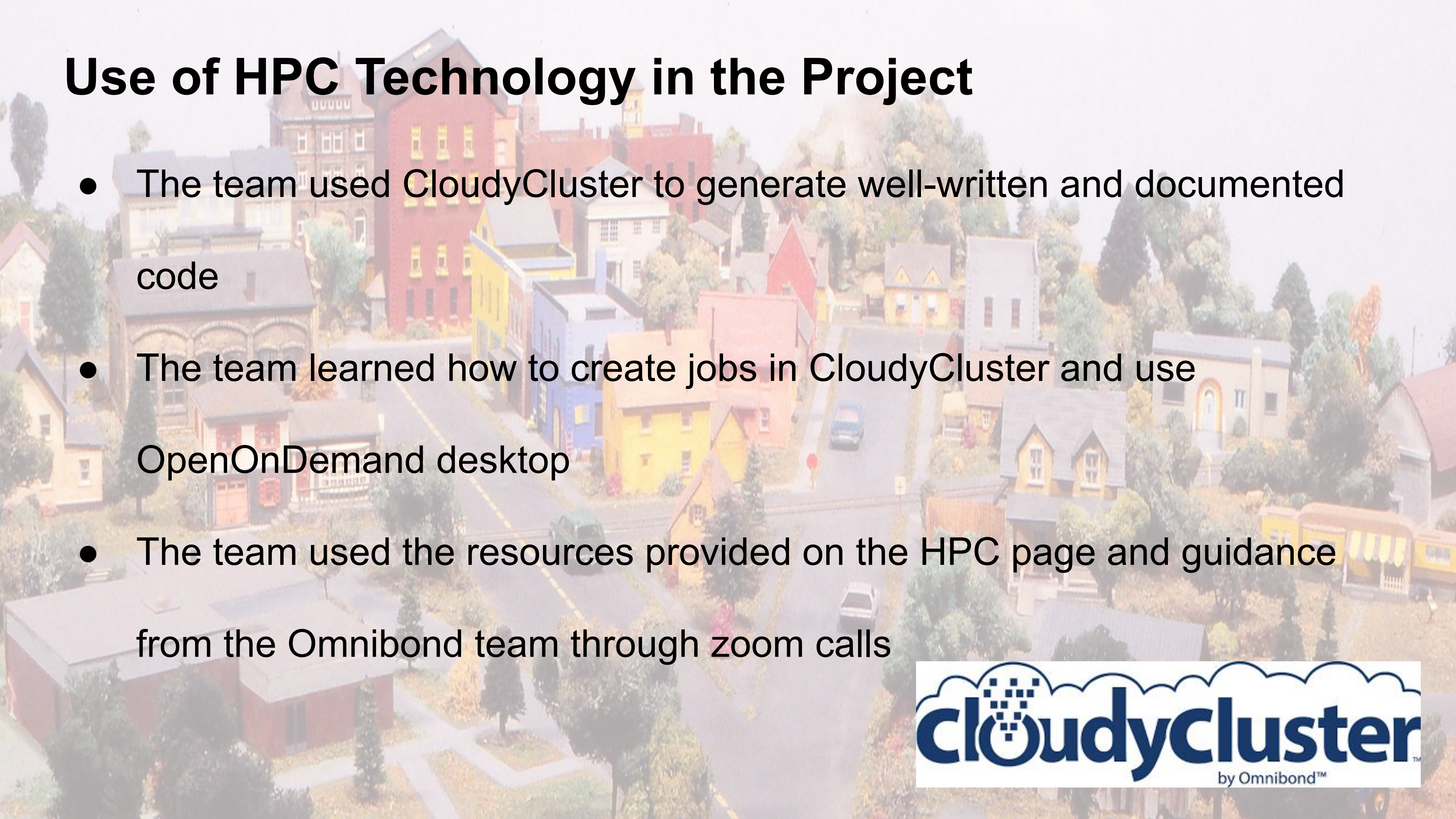
Michelle Ishimwe:

- Research and coding implementation, cloudycluster setup
- READme and presentation

Mentors: Edgar G, Pat S, Adonay S

- Continuous check-in calls and support to the team
- Provision of resources needed by the team

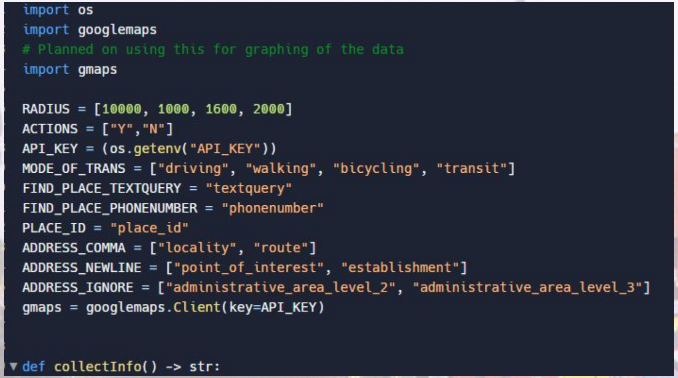
Use of HPC Technology in the Project



- The team used CloudyCluster to generate well-written and documented code
- The team learned how to create jobs in CloudyCluster and use OpenOnDemand desktop
- The team used the resources provided on the HPC page and guidance from the Omnibond team through zoom calls

cloudyCluster™ by Omnibond™

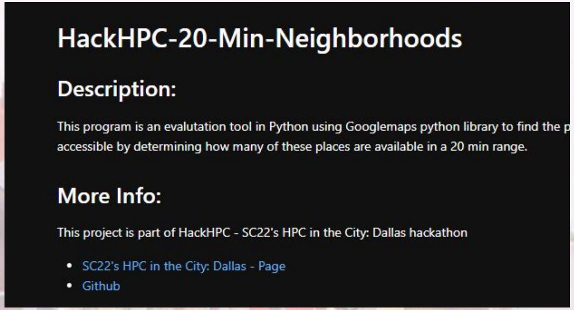
Demo Time



```
import os
import googlemaps
# Planned on using this for graphing of the data
import gmaps

RADIUS = [10000, 1000, 1600, 2000]
ACTIONS = ["Y","N"]
API_KEY = (os.getenv("API_KEY"))
MODE_OF_TRANS = ["driving", "walking", "bicycling", "transit"]
FIND_PLACE_TEXTQUERY = "textquery"
FIND_PLACE_PHONENUMBER = "phonenumber"
PLACE_ID = "place_id"
ADDRESS_COMMA = ["locality", "route"]
ADDRESS_NEWLINE = ["point_of_interest", "establishment"]
ADDRESS_IGNORE = ["administrative_area_level_2", "administrative_area_level_3"]
gmaps = googlemaps.Client(key=API_KEY)


def collectInfo() -> str:
```



HackHPC-20-Min-Neighborhoods

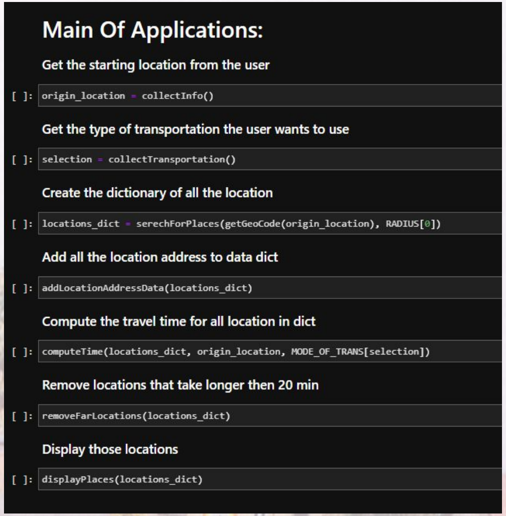
Description:

This program is an evalutation tool in Python using Googlemaps python library to find the p
accessible by determining how many of these places are available in a 20 min range.

More Info:

This project is part of HackHPC - SC22's HPC in the City: Dallas hackathon

- SC22's HPC in the City: Dallas - Page
- Github



Main Of Applications:

Get the starting location from the user

```
[ ]: origin_location = collectInfo()
```

Get the type of transportation the user wants to use

```
[ ]: selection = collectTransportation()
```

Create the dictionary of all the location

```
[ ]: locations_dict = serechForPlaces(getGeoCode(origin_location), RADIUS[0])
```

Add all the location address to data dict

```
[ ]: addLocationAddressData(locations_dict)
```

Compute the travel time for all location in dict

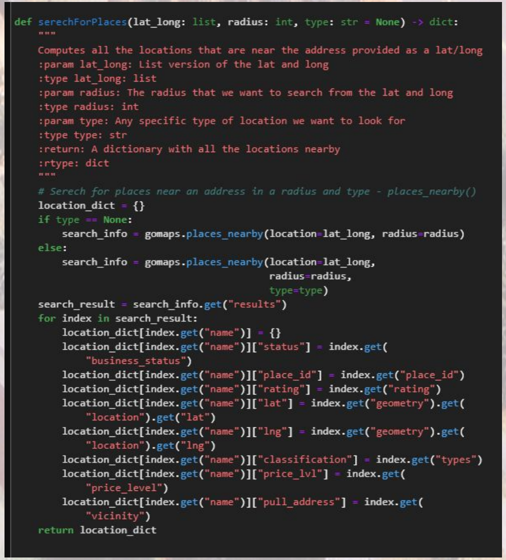
```
[ ]: computeTime(locations_dict, origin_location, MODE_OF_TRANS[selection])
```

Remove locations that take longer then 20 min

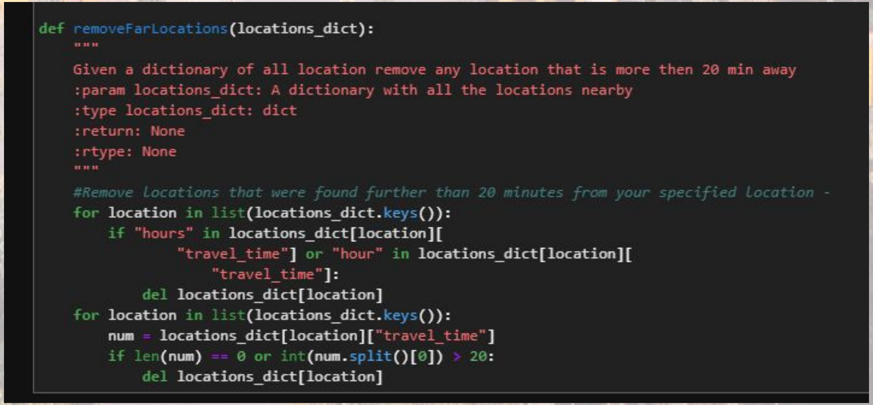
```
[ ]: removeFarLocations(locations_dict)
```

Display those locations

```
[ ]: displayPlaces(locations_dict)
```



```
def serechForPlaces(lat_long: list, radius: int, type: str = None) -> dict:
    """
    Computes all the locations that are near the address provided as a lat/long
    :param lat_long: List version of the lat and long
    :type lat_long: list
    :param radius: The radius that we want to search from the lat and long
    :type radius: int
    :param type: Any specific type of location we want to look for
    :type type: str
    :return: A dictionary with all the locations nearby
    :rtype: dict
    """
    # Serech for places near an address in a radius and type - places_nearby()
    location_dict = {}
    if type == None:
        search_info = gomaps.places_nearby(location=lat_long, radius=radius)
    else:
        search_info = gomaps.places_nearby(location=lat_long,
                                           radius=radius,
                                           type=type)
    search_result = search_info.get("results")
    for index in search_result:
        location_dict[index.get("name")] = {}
        location_dict[index.get("name")]["status"] = index.get(
            "business_status")
        location_dict[index.get("name")]["place_id"] = index.get("place_id")
        location_dict[index.get("name")]["rating"] = index.get("rating")
        location_dict[index.get("name")]["lat"] = index.get("geometry").get(
            "location").get("lat")
        location_dict[index.get("name")]["lng"] = index.get("geometry").get(
            "location").get("lng")
        location_dict[index.get("name")]["classification"] = index.get("types")
        location_dict[index.get("name")]["price_lvl"] = index.get(
            "price_level")
        location_dict[index.get("name")]["pull_address"] = index.get(
            "vicinity")
    return location_dict
```

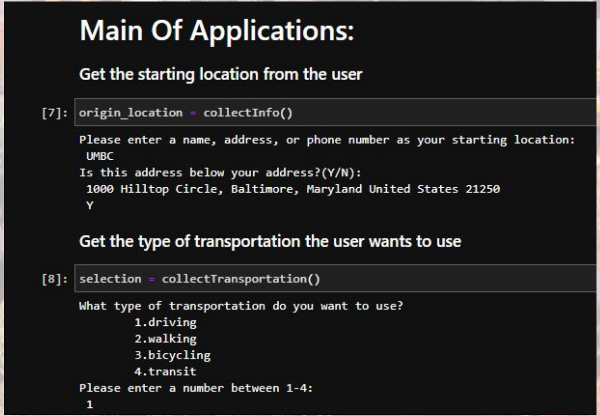


```
def removeFarLocations(locations_dict):
    """
    Given a dictionary of all location remove any location that is more then 20 min away
    :param locations_dict: A dictionary with all the locations nearby
    :type locations_dict: dict
    :return: None
    :rtype: None
    """
    #Remove locations that were found further than 20 minutes from your specified location -
    for location in list(locations_dict.keys()):
        if "hours" in locations_dict[location][
                "travel_time"] or "hour" in locations_dict[location][
                    "travel_time"]:
            del locations_dict[location]
    for location in list(locations_dict.keys()):
        num = locations_dict[location]["travel_time"]
        if len(num) == 0 or int(num.split()[0]) > 20:
            del locations_dict[location]
```

Community Impact of the Project

- If you were able to determine the accessibility of your neighborhood, how would your outlook on life change?
  - Would you take more walks?
  - Demand for the sidewalks to be fixed?
  - Vote for candidates that care about climate and land-use?

Don’t wait!! Use our evaluator tool!



Main Of Applications:

Get the starting location from the user

```
[7]: origin_location = collectInfo()
```

```
Please enter a name, address, or phone number as your starting location:
 UMBC
Is this address below your address?(Y/N):
 1000 Hilltop Circle, Baltimore, Maryland United States 21250
 Y
```

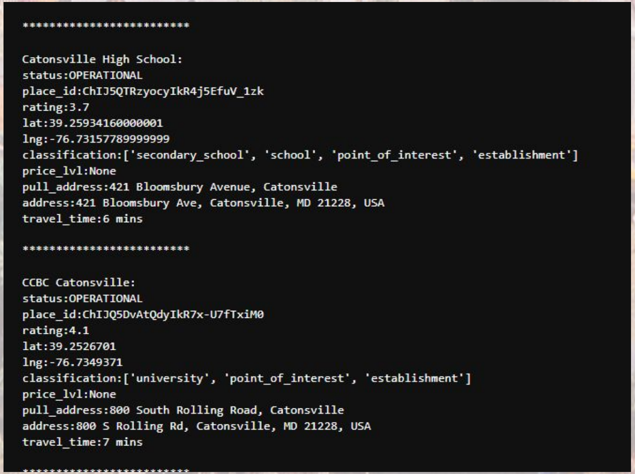
Get the type of transportation the user wants to use

```
[8]: selection = collectTransportation()
```

```
What type of transportation do you want to use?
        1.driving
        2.walking
        3.bicycling
        4.transit
Please enter a number between 1-4:
 1
```

What Next

- On HPC use Multiprocessing and Multithreading in python to help progress large amount of data
- Develop ways to display all data collect onto a user friendly map
- The ability to search for categorized locations
- Develop two user experience user interface for to types of target audiences
  - Everyday User:
    - People like you and me
    - Want to know what's accessible 20 min from them
  - Policy Makers and Researcher:
    - Quarry large amounts of data
    - Understand the communities they serve
    - Services that communities lack



```
*************************

Catonsville High School:
status:OPERATIONAL
place_id:ChIJ5QTRzyocyIkR4j5EfuV_1zk
rating:3.7
lat:39.25934160000001
lng:-76.73157789999999
classification:['secondary_school', 'school', 'point_of_interest', 'establishment']
price_lvl:None
pull_address:421 Bloomsbury Avenue, Catonsville
address:421 Bloomsbury Ave, Catonsville, MD 21228, USA
travel_time:6 mins

*************************

CCBC Catonsville:
status:OPERATIONAL
place_id:ChIJQ5DvAtQdyIkR7x-U7fTxiM0
rating:4.1
lat:39.2526701
lng:-76.7349371
classification:['university', 'point_of_interest', 'establishment']
price_lvl:None
pull_address:800 South Rolling Road, Catonsville
address:800 S Rolling Rd, Catonsville, MD 21228, USA
travel_time:7 mins

*************************
```

Thank You