Python_basics

November 15, 2024

[37]:
```
#create custom functions
def myfunction():
    print("hello I am a function")
myfunction()

#function with paramters
def myParameterFunction(list1):
    for item in list1:
        print(item);
myParameterFunction(["Apple","Orange","banana"])

#function with return type
def percentageCalculator(marks):
    percentages=[]
    for items in marks:
        percentage = round((items/100)*100,2)
        percentages.append(percentage)
    return percentages
percentageCalculator([100,86,56,25])
```

```
hello I am a function
Apple
Orange
banana
```

[37]:
```
[100.0, 86.0, 56.0, 25.0]
```

[35]:
```
#return function 2
def factorial(number):
    factorial =1
    while number > 0:
        factorial = factorial * (number)
        number= number -1
    return factorial
factorial(5)
print("the factorial is",factorial(5))
```

```
the factorial is 120
```

[39]:
```
def recfactorial(number):
    if number <= 1:
        return 1
    else:
        return number * factorial(number - 1)
print("The factorial of 5 is", factorial(5))
```

```
The factorial of 5 is 120
```

[23]:
```
#lambda functions are anonymous function
#they can be assigned to variable
#can contain any number of arguments but only one expression

#function to find power of a number
x= lambda a : pow(a,a)
print(x(5))

#lambda filter function
#syntax: filter(Predicate/condition,iterable)
#returns true or false based on condition
numbers=[1,2,3,4,5,6,7,8,9,10]
filtered_list = filter(lambda x: x % 2 == 0,numbers)
print(list(filtered_list))

#lambda reduce
#used to reduce the whole list to single cumulative value. doesnt return list
from functools import reduce
numbers=[1,2,3,4,5]
factorial = reduce(lambda x,y: x*y,numbers)
print('Factorial',factorial)

#map applies a given operation to every element in list
numbers=[1,2,3,4,5,6,7,8,9,10]
squared_list = list(map(lambda x: x*x ,numbers))
print('squared_list',squared_list)
```

```
3125
[2, 4, 6, 8, 10]
Factorial 120
squared_list [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

[42]:
```
#numpy functions
import numpy as np
#creating 3d arrays
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])
print(arr)
#1.Print the reverse NumPy array with type float.
```

```
list= [1, 2, 3, 4, -8, -10]
reversed_list = np.flip(list)
#2.converting integer array to float
float_list = np.array(reversed_list,dtype = 'float')
print(float_list)

#3.arithmetic operations on array
a = np.array([1, 2, 3, 4, 5])
b = np.array([10, 20, 30, 40, 50])
addition = a + b
subtraction = a - b
multiplication = a * b
division = a / b
#m*n and n*p result will be m*p
#first row* first column/ then first row second column then proceeding with␣
 ↪next rows with same columns
#in 1D array 10*1+20*2..
matrix_multiplication = np.dot(a,b)
print("Addition: ", addition)
print("Subtraction: ", subtraction)
print("Multiplication: ", multiplication)
print("Division: ", division)
print("matrix_multiplication",matrix_multiplication)

#array slicing
matrix =np.array([[1,2,3],[4,5,6],[7,8,9]])
#slice syntax : array[startrow:endrow, startcolumn:endcolumn]
#first row
first_row= matrix[0,:]
#last column
last_column = matrix[:,len(matrix)-1]
#middle elements in each row
#right index is not included
middle_elements = matrix[:,1:2]
print("first_row",first_row)
print("last_column",last_column)
print("middle_elements",middle_elements)

#array reshape
a=np.array([1,2,3,4,5,6,7,8,9,10,11,12])
array_2d = a.reshape(3,4)
array_flatten = array_2d.reshape(-1)
print("array_2d",array_2d)
print("array_flatten",array_flatten)
```

```
[[[1 2 3]
  [4 5 6]]
```

```
 [[1 2 3]
  [4 5 6]]]
[-10.  -8.   4.   3.   2.   1.]
Addition:  [11 22 33 44 55]
Subtraction:  [ -9 -18 -27 -36 -45]
Multiplication:  [ 10  40  90 160 250]
Division:  [0.1 0.1 0.1 0.1 0.1]
matrix_multiplication 550
first_row [1 2 3]
last_column [3 6 9]
middle_elements [[2]
 [5]
 [8]]
array_2d [[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
array_flatten [ 1  2  3  4  5  6  7  8  9 10 11 12]
```

[10]:
```python
#string functions
#1.swap case
sentence = "iLOVEpROGRAMing"
print(sentence.swapcase())

#2. Split the string on a " " (space) delimiter and join using a - hyphen.
sentence = "i LOVE pROGRAMing"
splitstring = sentence.split(' ')
joinedstring = '-'.join(splitstring)
print(joinedstring)

#3.string validation
input_string = "qA2"
for char in input_string:
    if char.isalnum():
        print("Contains alpha and numeric",char)
    if char.isdigit():
        print("contains numbers",char)
    if char.islower():
        print("contains lowercase",char)
    if char.isupper():
        print("contains uppercase",char)
#4.replace string function
a = "Hello, World!"
print(a.replace("H", "J"))
```

```
IloveProgramING
i-LOVE-pROGRAMing
Contains alpha and numeric q
```

```
contains lowercase q
Contains alpha and numeric A
contains uppercase A
Contains alpha and numeric 2
contains numbers 2
Jello, World!
```

[21]:
```python
#for loops
#1.iterating through string
sentence = "python"
for char in sentence:
    print(char)
#2.looping using given range
for i in range(2):
    print(i)
#3.looping with increment
for i in range(1,10,3):
    print(i)
#4.reverse of a string
for char in range(len(sentence) - 1, -1, -1):
    print('reverse of a a string',sentence[char])
#5.fibonacci series
def fibnocci(number):
    a=0
    b=1
    list=[]
    while len(list)<number:
        list.append(a)
        a,b=b,a+b
    print(list)
fibnocci(5)
```

```
p
y
t
h
o
n
0
1
1
4
7
reverse of a a string n
reverse of a a string o
reverse of a a string h
```

```
reverse of a a string t
reverse of a a string y
reverse of a a string p
[0, 1, 1, 2, 3]
```

[50]:
```python
#tuples
#immutbale-no appending,deleting or insetion

#1.tuple creation
my_tuple = (1,2,"hello",3,4)
#2.accessing subset of tuples
subset_tuple = my_tuple[0:3]
print(subset_tuple)
#2 and 4 count of eoccurances and index of elements
print(my_tuple.count("hello"))
print(my_tuple.index(4))

#5Question: Tuple Operations and Manipulation
#You are given a list of tuples, where each tuple contains the following␣
 ↪structure:
#The first element is the name of a student (string).
#The second element is their grade (integer).
#Task:
#Find the highest grade in the list and return it.
#Find the lowest grade in the list and return it.
#Calculate the average grade of all students and return it, rounded to two␣
 ↪decimal places


def gradeFinder(input_tuple):
    highest_grade=0
    lowest_grade=100
    total_grade =0
    average =0
    for name,grade in input_tuple:
        if grade > highest_grade:
            highest_grade = grade
    print('highest_grade',highest_grade)
    for name,grade in input_tuple:
        if grade < lowest_grade:
            lowest_grade = grade
    print('lowest_grade',lowest_grade)
    for name,grade in input_tuple:
        total_grade = total_grade+grade
    average =total_grade/len( input_tuple)
    print(average)
```

```
input_tuple = [("Alice", 85), ("Bob", 92), ("Charlie", 78), ("David", 88),␣
 ↪("Eva", 91)]
gradeFinder(input_tuple)
```

```
(1, 2, 'hello')
1
4
highest_grade 92
lowest_grade 78
86.8
```

[101]:
```
#1.dictionary practice question
def count_word_frequency(text):
    #to convert into lower case
    lower_text = text.lower()
    #to remove punctutaions
    for punctuations in ('!',','):
       lower_text= lower_text.replace(punctuations,"")
    #to split into words
    split_list= lower_text.split(' ')
    print(split_list)
    #empty dictionary
    dict={}
    for words in split_list:
        if words in dict:
            dict[words]+=1
        else:
            dict[words]=1
    #results
    print("word frequency",dict)
text = "Hello hello world, this is a world of python. Hello, world!"
count_word_frequency(text)
#dictionary practice question2
#find character frequency
def character_frequency(text):
    lower_text = text.lower()
    #remove punctuations:
    for punctuations in (',','!'):
      lower_text=  lower_text.replace(punctuations,"");
    dict={}
    for char in lower_text:
        if char in dict:
            dict[char]+=1
        else:
            dict[char]=1
    print("character frequency",dict)
```

```
text = "Hello, World!"
character_frequency(text)
#inventory tracker
def inventory_tracker(sales_data):
    dict={}
    for item,quantity in sales_data:
        if item in dict:
            dict[item]+=quantity
        else:
            dict[item]=quantity
    print(dict)
sales_data = [
    ("apple", 3),
    ("banana", 5),
    ("apple", 2),
    ("orange", 4),
    ("banana", 1),
    ("apple", 1)
]
inventory_tracker(sales_data)
```

```
['hello', 'hello', 'world', 'this', 'is', 'a', 'world', 'of', 'python.',
'hello', 'world']
word frequency {'hello': 3, 'world': 3, 'this': 1, 'is': 1, 'a': 1, 'of': 1,
'python.': 1}
character frequency {'h': 1, 'e': 1, 'l': 3, 'o': 2, ' ': 1, 'w': 1, 'r': 1,
'd': 1}
{'apple': 6, 'banana': 6, 'orange': 4}
```

[115]:
```
#arithemetic functions
#1.bmi calculator
def calculatebmi(weight,height):
    BMI = weight/(height*height)
    print(round(BMI,2))
    if BMI < 18.5:
        print("Under weight")
    elif 18.5 <= BMI < 24.9:
        print("Normal Weight")
    elif 25 <= BMI < 29.9:
        print("Over weight")
    else:
        print("Obesity")
calculatebmi(70,1.75)
```

```
22.86
Normal Weight
```

[55]:
```
pip install openpyxl
```

```
Defaulting to user installation because normal site-packages is not writeable
Looking in links: /usr/share/pip-wheels
Requirement already satisfied: openpyxl in
/opt/conda/envs/anaconda-2022.05-py39/lib/python3.9/site-packages (3.0.9)
Requirement already satisfied: et-xmlfile in
/opt/conda/envs/anaconda-2022.05-py39/lib/python3.9/site-packages (from
openpyxl) (1.1.0)
Note: you may need to restart the kernel to use updated packages.
```

[57]:
```
import pandas as pd
```

[81]:
```
dframe = pd.read_excel('/home/c1caac95-76bb-4d34-a649-06ed0d881be7/
 ↪Python_practice/Dim_category.xlsx')
print(dframe.head(5))
print(dframe.columns)
#rename columns
dframe = dframe.rename(columns={"Category_id.1": "Category_description"})
print(dframe.columns)
#unique category
unique_categories = dframe['Category_description'].unique()
print(unique_categories)
```

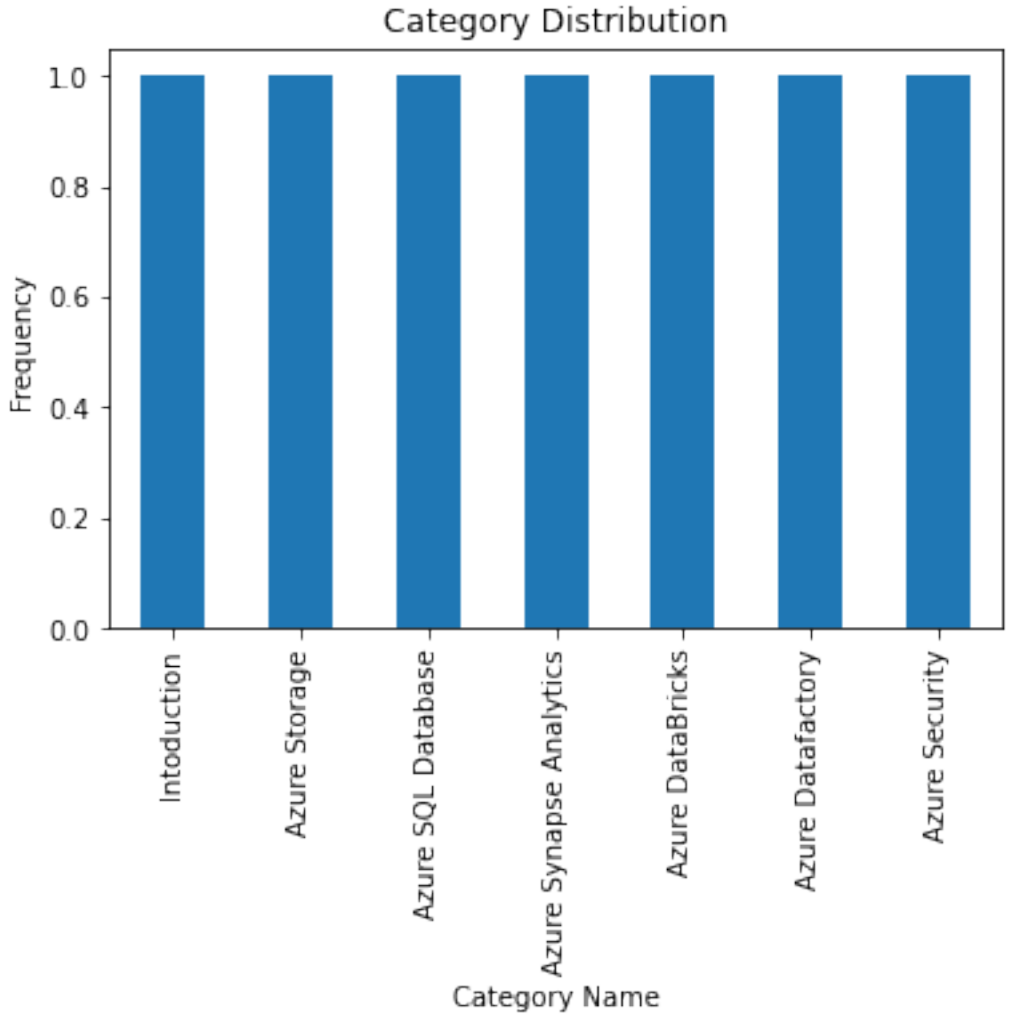
```
   Category_id            Category_id.1
0            1              Intoduction
1            2            Azure Storage
2            3       Azure SQL Database
3            4  Azure Synapse Analytics
4            5         Azure DataBricks
Index(['Category_id', 'Category_id.1'], dtype='object')
Index(['Category_id', 'Category_description'], dtype='object')
['Intoduction' 'Azure Storage' 'Azure SQL Database'
 'Azure Synapse Analytics' 'Azure DataBricks' 'Azure Datafactory'
 'Azure Security']
```

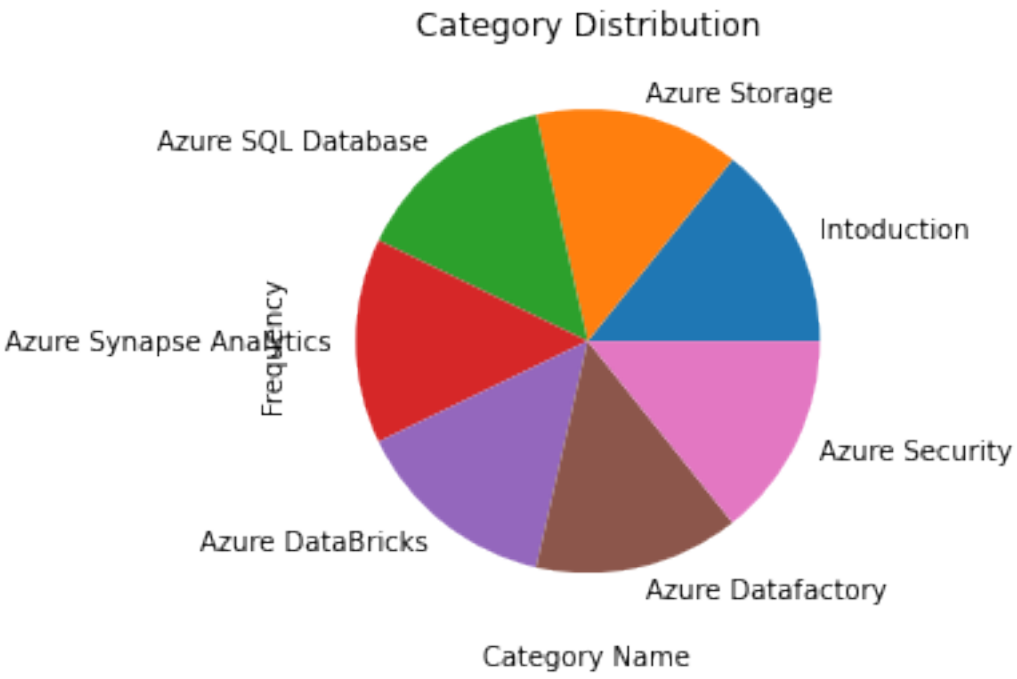
[69]:
```
print(os.getcwd())
```

```
/home/c1caac95-76bb-4d34-a649-06ed0d881be7/Python_practice
```

[87]:
```
import matplotlib.pyplot as plt

dframe['Category_description'].value_counts().plot(kind='bar')

plt.title('Category Distribution')
plt.xlabel('Category Name')
plt.ylabel('Frequency')
```

```
plt.show()

dframe['Category_description'].value_counts().plot(kind='pie')

plt.title('Category Distribution')
plt.xlabel('Category Name')
plt.ylabel('Frequency')
plt.show()
```




Category Distribution
Azure Storage
Azure SQL Database
Intoduction
Azure Synapse Analytics
Frequency
Azure Security
Azure DataBricks
Azure Datafactory
Category Name

[ ]: