{

"cells": [

{

"cell\_type": "code",

"execution\_count": 2,

"id": "0e6f2909",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"sorted list = [19, 19, 20, 22, 24, 24, 24, 25, 25, 26]\n",

"max age:26, min age:19\n",

"added list: [19, 19, 20, 22, 24, 24, 24, 25, 25, 26, 19, 26]\n",

"6\n",

"The median of ages: 24\n",

"Average age is: 22.75\n",

"Range of the ages: 7\n"

]

}

],

"source": [

"#QUESTION 1\n",

"\n",

"ages = [19, 22, 19, 24, 20, 25, 26, 24, 25, 24] ##creating a list for the ages\n",

"\n",

"ages.sort() #sorting the Ages list\n",

"print(\"sorted list = \", ages)\n",

"\n",

"print(f\"max age:{max(ages)}, min age:{min(ages)}\") #To print the min and max age\n",

"ages.extend([min(ages),max(ages)]) #adding min and max ages to the ages list\n",

"print(f\"added list: {ages}\") # list after addition\n",

"\n",

"Middle\_Index = int(len(ages)/2) # middle index in the list\n",

"print(Middle\_Index)\n",

"if Middle\_Index % 2 == 0: #Condition to find if the length is even\n",

" print(f\"The median of ages: {int((ages[Middle\_Index-1] + ages[Middle\_Index])/ 2)}\") #Printing the median age\n",

"\n",

"print(f\"Average age is: {sum(ages)/len(ages)}\") #Printing the average age\n",

"\n",

"print(f\"Range of the ages: {max(ages) - min(ages)}\") # printing the range of the ages \n"

]

},

{

"cell\_type": "code",

"execution\_count": 1,

"id": "0df255ac",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"length of student: 9\n",

"<class 'list'> ['Critical Thinking ']\n",

"['Critical Thinking ',Hardworking, 'Creativity', 'Leadership']\n",

"keys of student dictionary:dict\_keys(['first\_name', 'last\_name', 'gender', 'age', 'martial status', 'skills', 'country', 'city', 'address'])\n",

"values of student dictionary: dict\_values(['ram', 'kethi', 'male', '23', 'Single', ['Critical Thinking ', 'Hardworking', 'Creativity', 'Leadership'], 'India', 'hyd', 'kmm'])\n"

}

],

"source": [

"#QUESTION 2\n",

"\n",

"dog = {} #Created a empty dictionary\n",

"dog = {\"name\":\"nicky\", \"color\":\"red\", \"bread\":\"Pitbull\", \"legs\":\"4\", \"age\":\"1\"} #adding keys and values to the created directory\n",

"\n",

"#Created a empty student dictionary\n",

"student= {} \n",

"student= {\"first\_name\":\"ram\",\"last\_name\":\"kethi\",\"gender\":\"male\",\"age\":\"23\",\"martial status\":\"Single\",\n",

" \"skills\":['Critical Thinking '],\"country\":\"India\",\"city\":\"hyd\",\"address\":\"kmm\"} #adding key values to Student Dictionary\n",

"\n",

"print(f\"length of student: {len(student)}\") # Printing the lenght of student dictionary\n",

"\n",

"print(type(student['skills']),student['skills']) #Printing the values in skills and type of date \n",

"\n",

"student['skills'].extend(['Hardworking', 'Creativity', 'Leadership']) #modifying skills by extend function\n",

"print(student['skills'])\n",

"\n",

"print(f\"keys of student dictionary:{student.keys()}\") #Printing dictionary keys as list\n",

"\n",

"print(f\"values of student dictionary: {student.values()}\") #Printing dictionary keys as values\n",

"\n",

" "

]

},

{

"cell\_type": "code",

"execution\_count": 2,

"id": "0230a025",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"no of siblings: 6\n",

"('SaiRam', 'Sathish', 'sravan, 'kitta, 'harshith', 'mahi')\n",

"type: <class 'tuple'>\n",

"family member: ('Sairam', 'Sathish', 'sravan', 'kitta', 'harshith', 'mahi', 'rao', 'veni')\n"

]

}

],

"source": [

"#QUESTION 3\n",

"\n",

"brothers = (\"SaiRam\", \"sravan\", \"harshith\") #Created a brother tuple\n",

"sisters = (\"bhagi\", \"Lasya\", \"desi\") #Created a sisters tuple\n",

"siblings = brothers+sisters #Adding the siblings tuple \n",

"print(f\"no of siblings: {len(siblings)}\") #Printing the number of siblings\n",

"print(siblings)\n",

"family\_member = () # created a family\_member tuple\n",

"family\_member += siblings + (\"rao\", \"veni`\") # modifying the family\_member tuple by adding sibling to it\n",

"print(\"type: \",type(family\_member)) # Printing the type of family\_member\n",

"print(\"family member: \",family\_member) "

]

},

{

"cell\_type": "code",

"execution\_count": 3,

"id": "dda44599",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"length of it\_companies: 7\n",

"add new company: {'IBM', 'Google', 'Apple', 'Oracle', 'Amazon', 'Microsoft', 'Facebook', 'twitter'}\n",

"updated it companies: {'TCS', 'ValueLabs', 'Accenture'}\n",

"remove it companies: {'ValueLabs'}\n",

"discard it companies: {'Cognizant'}\n",

"join A union B:{19, 20, 22, 24, 25, 26, 27, 28}\n",

"join A intersection B:{19, 20, 22, 24, 25, 26}\n",

"if A is subset of B:True\n",

"if A is disjoint of B:False\n",

"{19, 20, 22, 24, 25, 26, 27, 28}\n",

"{19, 20, 22, 24, 25, 26, 27, 28}\n",

"symmetric difference:set()\n",

"deleting both A and B: None None\n",

"The length of the list age:8\n",

"The length of the set age:8\n"

]

}

],

"source": [

"#QUESTION 4\n",

"\n",

"it\_companies = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'}\n",

"A = {19, 22, 24, 20, 25, 26}\n",

"B = {19, 22, 20, 25, 26, 24, 28, 27}\n",

"age = [22, 19, 24, 25, 26, 24, 25, 24]\n",

"count = len(it\_companies)\n",

"print(f\"length of it\_companies: \", count) #Printing the length of companies\n",

"\n",

"it\_companies.add(\"twitter\") #add twitter to the it\_companies Set\n",

"\n",

"print(f\"add new company: \", it\_companies)\n",

"it\_companies.update(['TCS', 'ValueLabs', 'Accenture']) # multiple it companies\n",

"print(f\"updated it companies:\", it\_companies) \n",

"\n",

"it\_companies.remove('ValueLabs') #remove one company(which raises error)\n",

"print(f\"remove it companies:\", it\_companies)\n",

"\n",

"it\_companies.discard('Cognizant') #discard which doesnt raise an error\n",

"print(f\"discard it companies:\", it\_companies)\n",

"\n",

"print(f\"join A union B:{A | B}\") # Printing the Union of A and B\n",

"print(f\"join A intersection B:{A & B}\") # Printing the intersection of A and B\n",

"print(f\"if A is subset of B:{A.issubset(B)}\") #Printing the subset of B \n",

"print(f\"if A is disjoint of B:{A.isdisjoint(B)}\") #printng the disjoint of B\n",

"\n",

"A.update(B) #joint A with B and B with A\n",

"print(A) \n",

"B.update(A)\n",

"print(B)\n",

"\n",

"print(f\"symmetric difference:{A.symmetric\_difference(B)}\") #Printing the symmetric difference\n",

"A.clear()\n",

"B.clear()\n",

"print(f\"deleting both A and B: \", A.clear(), B.clear()) #Printing the deleted set \n",

"print(f\"The length of the list age:{len(age)}\")\n",

"AgeSet=set(age) #convert age to set and compare length of list and set\n",

"print(f\"The length of the set age:{len(age)}\")\n"

]

},

{

"cell\_type": "code",

"execution\_count": 4,

"id": "435d5e39",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"radius of the circle is: 30\n",

"area of circle: 2826.0\n",

"circumference of circle: 188.4\n",

"enter radius25\n",

"Area from user input radius: 1962.5\n"

]

}

],

"source": [

"#QUESTION 5\n",

"\n",

"radius=30 #taking radius as per input\n",

"print(f\"radius of the circle is: \", radius)\n",

"area\_of\_circle=3.14\*(radius\*\*2) #area of circle \n",

"circumference\_of\_circle=2\*3.14\*radius #circumference of circle\n",

"print(f\"area of circle: \", area\_of\_circle)\n",

"print(f\"circumference of circle: \", circumference\_of\_circle)\n",

"radius\_input = int(input(\"enter radius\")) #To take the input from the console\n",

"new\_area = 3.14\*radius\_input\*radius\_input\n",

"print(f\"Area from user input radius: \", new\_area)\n"

]

},

{

"cell\_type": "code",

"execution\_count": 5,

"id": "eceac2f4",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"['I', 'am', 'a', 'teacher', 'and', 'I', 'love', 'to', 'inspire', 'and', 'teach', 'people']\n",

"number of unique words in the given string: 10\n"

]

}

],

"source": [

"#QUESTION 6\n",

"\n",

"sentence = \"I am a teacher and I love to inspire and teach people\"\n",

"list = sentence.split(' ') #Split function by default uses whiteSpaces as the separator in the string sentence\n",

"print(list)\n",

"unique\_words=len(set(list)) #number of unique words in given string\n",

"print(f\"number of unique words in the given string: \",unique\_words ) # printing the Unique\_words length"

]

},

{

"cell\_type": "code",

"execution\_count": 6,

"id": "ba50a302",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Name\t\tAge\t\tCountry\t\tCity\n",

"Asabeneh\t250\t\tFinland\t\tHelsinki\n"

]

}

],

"source": [

"# QUESTION 7\n",

"\n",

"print(\"Name\\t\\tAge\\t\\tCountry\\t\\tCity\\nAsabeneh\\t250\\t\\tFinland\\t\\tHelsinki\") #To print the text by using the tab escape sequence\n"

]

},

{

"cell\_type": "code",

"execution\_count": 7,

"id": "de690597",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"The area of a circle with radius 10 is 314.0 meters square.\n"

]

}

],

"source": [

"#QUESTION 8\n",

"\n",

"radius = 10\n",

"area = 3.14 \* radius \*\* 2\n",

"print(f\"The area of a circle with radius {radius} is {area} meters square.\") #To find the radius of the circle\n",

"\n"

]

},

{

"cell\_type": "code",

"execution\_count": 10,

"id": "ebe6b229",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"weight in kgs: [68.0388, 70.30676, 65.77083999999999, 67.131616]\n"

]

}

],

"source": [

"#QUESTION 9\n",

"\n",

"#list of students weights\n",

"weight\_lbs = [150, 155, 145, 148]\n",

"weight\_kgs = []\n",

"\n",

"#converting weight to kgs\n",

"for x in weight\_lbs:\n",

" weight\_kgs.append(x\*0.453592)\n",

"print(f\"weight in kgs: \",weight\_kgs) # Printing the Weight in Kgs"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "a63093cd",

"metadata": {},

"outputs": [],

"source": []

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3 (ipykernel)",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.11.5"

}

},

"nbformat": 4,

"nbformat\_minor": 5

}