mbctjurjq

January 23, 2025

[ ]:
```
dct={} #declaring a empty dictionary type 1
data=dict() # using  a function
```

[ ]:
```
dct={}
dct['name'] ='raman'
print(dct)
```

```
{'name': 'raman'}
```

[ ]:
```
dct=dict([("name","raman"),("address","ktm")]) #another method to create a dictionary
dct["500"]="hari" # supports numerical value name
dct
```

[ ]:
```
{'name': 'raman', 'address': 'ktm', '500': 'hari'}
```

[ ]:
```
data['route']=['abc','def',[1,2,3,[7,8]]]
data
```

[ ]:
```
{'route': ['abc', 'def', [1, 2, 3, [7, 8]]]}
```

[ ]:
```
#update function in dictionary
dct['name']='hari'
print(dct)
```

```
{'name': 'hari', 'address': 'ktm', '500': 'hari'}
```

[ ]:
```
#deleting a value in dictionary
del dct['name']
dct
```

[ ]:
```
{'address': 'ktm', '500': 'hari'}
```

[ ]:
```
# getting keys and values from a dictionary
print(dct.keys())
print(dct.values())
```

```
dict_keys(['address', '500'])
dict_values(['ktm', 'hari'])
```

[ ]:

```
#getting a list of the keys and values
keys_list=dct.keys() # extracting keys and converting to list
print(keys_list)
values_list=dct.values() # extracting values and converting to list
print(values_list)
```

[ ]:

```
#nested dictionary
classroom={}
classroom['student1']={"name":"Harry","Address": "KTM","school": "Ku"}
classroom['student2']={"name":"gopal","Address": "bkt","school": "tu"}
classroom
```

[ ]:

```
{'student1': {'name': 'Harry', 'Address': 'KTM', 'school': 'Ku'},
 'student2': {'name': 'gopal', 'Address': 'bkt', 'school': 'tu'}}
```

[ ]:

```
#operators in python
#1. Arithmetic
a=7
b=2
print(a+b) #addition
print(a-b) #subtraction
print(a*b) #multiplication
print(a/b) #division
print(a%b) #mod
print(a**b) #power
print(a//b) #floor division
```

```
9
5
14
3.5
1
49
3
```

[ ]:

```
#assignment operators
a=7 # normal assignment
a+=1 # equivalent to a=a+1
a-=1 #equivalent to a=a-1
```

[ ]:

```
# bit wise operator
x=10
y=4
print(x&y) # bitwise and
print(x|y) #bitwise or
print(x^y)  #bitwise xor
print(~x) #bitwise not
```

[ ]:

```
#identiy operator
x1=[6,7,8]
x2=[1,2,3]
print(x1 is not x2)
print(x1 is x2)
```

```
True
False
```

[ ]:

```
#interesting fact
xc=255
yc=255
print(xc is yc) #returns true
xc1=257
yc1=257
print(xc1 is yc1) #returns false
```

```
True
False
```

[ ]:

```
#membership operator
data="helo"
print("h" in data)
print("h" not in data)
```

```
True
False
```

[ ]:

```
#membership operator in
a="I love python"
if 'love' in a:
  print("yes")
else:
  print("no")
```

```
yes
```

[ ]:

```
# program to check whether a number is prime or not
def check_prime(num):
  if num==1:
    print("Number is neither prime nor composite")
  else:
    for i in range(1,int(num/2)):
      if num%i==0:
```

```
    print("Number is composite")
    break
  else:
   print("Number is prime")
check_prime(4)
```

```
Number is composite
```

[ ]:

```
def recursion(num): #factorial using recursion
 if num==1:
  return 1
 else:
  return num*recursion(num-1)

recursion(4)
```

[ ]: 24

[ ]:

```
#conditional Statements and fucntions
def calculat()->int:
 hour =1
 minute=2
 return (hour,minute)

x,y=calculat()
print(x,y)
```

```
1 2
```

[ ]:

```
#profit or loss
def finance(sp,cp):
 if(sp>cp):
  print("Profit")
  print("Profit percent= ",((sp-cp)/cp)*100)
 elif(sp==cp):
  print("break even")
 else:
  print("Loss")
  print("Loss percent= ",((cp-sp)/cp)*100)
a=int(input("ENter the sp"))
b=int(input("ENter the cp"))
finance(a,b)
```

```
ENter the sp100
ENter the cp50
Profit
Profit percent=  100.0
```