python by zen

In [1]:

```
import this
```

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

operation,operator and oprand

In [*]:

```
5 + 7 #operation
# + operator
# 5,7 oprand
#PEMDAS
print(8*7+3/2-4) #output
#56+3/2-4
#56+1.5-4
#57.5-4
#53.5
```

how python will store his data

In [4]:

```
a = 8
b = 8
print(a)
print(b)
```

```
8
8
```

In [5]:

```
print(id(a))
print(id(b))
```

```
140714910327408
140714910327408
```

how to find the type of data.......by function type

In [2]:

```
a = 8
type(a)
```

Out[2]:

```
int
```

In [3]:

```
a = 'hamza'
type(a)
```

Out[3]:

```
str
```

difference between list and tuple......only barcket difference

In [6]:

```
#index      0       1    2    3     4
names =("hamza","ali",20,True,False)
#index     -5      -4   -3    -2    -1
type(names)     #bracket difference
```

Out[6]:

```
tuple
```

In [7]:

```
#index      0       1    2    3     4
names =["hamza","ali",20,True,False]
#index     -5      -4   -3    -2    -1
type(names)      #bracket difference
```

Out[7]:

```
list
```

In [8]:

```
print(names)
print(names[1])
```

```
['hamza', 'ali', 20, True, False]
ali
```

In [9]:

```
names = ("hamza","google","hamza")
print(names)
type(names)
```

```
('hamza', 'google', 'hamza')
```

Out[9]:

```
tuple
```

In [10]:

```
names = {"hamza","google","hamza"}
print(names)
print(type(names))
names = list(names)
print(names[1])
type(names)
```

```
{'hamza', 'google'}
<class 'set'>
google
```

Out[10]:

```
list
```

PERFORMING SLICING BY 'dir function'.........important part of microsoft test

In [ ]:

```
dir(name)          #ek function jo kise chez ka tamam attributes apko dee daaa ga function=
```

In [11]:

```
name1 ="muhamMad qAsIm"
print(name1.lower())                 #inline perform hoo raha ha function
print(name1)
```

```
muhammad qasim
muhamMad qAsIm
```

In [12]:

```
name1 = "     MUHAmmad QasIm     "
print(len(name1))
print(name1.strip())
print(len(name1.strip()))
```

```
24
MUHAmmad QasIm
14
```

In [13]:

```
a = "we are pakistan we love our country"
a.find("pakistan")
a[a.find("we"):5]                      #we se aga 5 words count kroo
b = a.split()
b                                     #phr hum es se aur function krwa skta hain
```

Out[13]:

```
['we', 'are', 'pakistan', 'we', 'love', 'our', 'country']
```

COCATENATION

bY SIMPLE METHOD

In [*]:

```
name = ["hamza"]
fname = ["muhammad"]
programe = "paic"
print("Student Name : name \nFather Name : fname \nPrograme: programe \n (name,fname,progra
```

BY PLUS METHOD

In [25]:

```
namez = "hamza"
fnamez = "konain"
programz = "piaic"

#+
print("student name : " + namez + "\n father name : " + fnamez + "\n programe : " + program
```

```
student name : hamza
 father name : konain
 programe : piaic
```

BY PERCENTAGE METHOD(IMPORTANT)

In [*]:

```
name = input("enter name")      #ENTER NAME WHEN EXECUTE THE CELL
fname = input("enter fname")    #ENTER FATHER NAME WHEN EXECUTE THE CELL
programe = "paic"
city = input("enter city name")#ENTER CITY NAME WHEN EXECUTE THE CELL
address = input("enter address")#ENTER ADDRESS EHRN EXECUTE THE CELL
country = "pakistan"
score = 30            #SUGEST ME HOW I CAN USE INPUT FUNCTION BY DEFINING NUM AND INTEGER...
print(" 1)Student Name: %s \n 2)Father Name: %s \n 3)Programe: %s \n 4)City: %s \n 5)Addres
7)score: %d"%(name,fname,programe,city,address,country,score))
```

```
enter name
```

In [ ]: