Given the string `s = "PYTHON"`, retrieve and print the following:

a) The first character

b) The last character

c) The string "H"

d) The `type` of "H"

In [10]:

```
s = 'python'
s[0]
s[5]
s[3]
type(s)
```

```
python
```

2. Indexing Tuples:

Given the tuple `t = (10, 20, 30, 40, 50, 60)`, retrieve and print the following:

a) The first element

b) The third element c) (Try to) Set the 3rd element to `30.4`

In [16]:

```
t = (10, 20, 30, 40, 50, 60)
t[0]
t[2]
t[2] = 30.4
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[16], line 4
      2 t[0]
      3 t[2]
----> 4 t[2] = 30.4

TypeError: 'tuple' object does not support item assignment
```

3. Indexing Lists and Lists of Lists:

Given the list `lst = [5, 15, 25, [35, 45, [55, 65, 75], 85], 95]`, retrieve and print the following:

a) The first element

b) The last element

c) The sublist `[55, 65, 75]` d) The number `65`

In [21]:

```
list = [5, 15, 25], [35, 45], [55, 65, 75]
list[0]
```

Out[21]:

```
[5, 15, 25]
```

In [22]:

```
list[2]
```

Out[22]:
```
[55, 65, 75]
```

In [23]:
```
list[2][1]
```

Out[23]:
```
65
```

In [24]:
```
list[2]
```

Out[24]:
```
[55, 65, 75]
```

4. Lists of Lists as Arrays:

Consider a 3x3 matrix represented as a list of lists:

```python matrix = [ [1, 2, 3], [4, 5, 6], [7, 8, 9]

a) Retrieve the second row.

b) Retrieve the third column.

c) Change the center element to 0 and print the modified matrix.

In [25]:
```
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
matrix[1]
```

Out[25]:
```
[4, 5, 6]
```

In [26]:
```
matrix[2]
```

Out[26]:
```
[7, 8, 9]
```

In [27]:
```
matrix[1] = 0
print(matrix)
```

```
[[1, 2, 3], 0, [7, 8, 9]]
```

5. Using Sets:

Given two lists `A = [1, 2, 2, 3, 4, 4, 5]` and `B = [4, 5, 5, 6, 7, 7, 8]` :

a) Create sets from both lists.

b) Find the union of the two sets.

c) Find the intersection of the two sets.

d) Find the elements that are in A but not in B. e) Find the elements that are in A or B but not both.

In [28]:
```
a = {1, 2, 2, 3, 4, 4, 5}
b = {4, 5, 5, 6, 7, 7, 8}
```
```

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

```
{1, 2, 3, 4, 5}
{4, 5, 6, 7, 8}
```

In [29]:

```
{1, 2, 2, 3, 4, 4, 5} == {4, 5, 5, 6, 7, 7, 8}
```

Out[29]:

```
False
```

In [30]:

```
a | b
```

Out[30]:

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [31]:

```
a & b
```

Out[31]:

```
{4, 5}
```

In [33]:

```
a - b
```

Out[33]:

```
{1, 2, 3}
```

In [34]:

```
a ^ b
```

Out[34]:

```
{1, 2, 3, 6, 7, 8}
```

6. Working with Dictionaries:

Consider the following dictionary that represents the stock of items in a store:

```
stock = {
    "apple": 50,
    "banana": 25,
    "orange": 30,
    "grape": 45
}
```

a) Retrieve the stock of `apple`.

b) Add a new fruit, `pear`, with a stock of 40.

c) Update the stock of `banana` to 30.

d) Remove `orange` from the stock.

In [36]:

```
stock = {
    "apple": 50,
    "banana": 25,
    "orange": 30,
    "grape": 45
}

print(stock['apple'])
```

```
50
```

In [37]:

```
stock['pear'] = 50
```

In [39]:

```
print(stock)
```

```
{'apple': 50, 'banana': 25, 'orange': 30, 'grape': 45, 'pear': 50}
```

In [40]:

```
stock['banana'] = 30
```

In [41]:

```
print(stock)
```

```
{'apple': 50, 'banana': 30, 'orange': 30, 'grape': 45, 'pear': 50}
```

In [42]:

```
del stock['orange']
print(stock)
```

```
{'apple': 50, 'banana': 30, 'grape': 45, 'pear': 50}
```

In [ ]:

```

```