ses 2 presentation if loops string

October 8, 2021

1 Condition

if statement

```
if condition:
    statements
```

Short Hand if statement

```
if condition: statement
```

Short Hand if-else statement

```
statement_when_True if condition else statement_when_False
```

if else statment

```
if (condition): Executes this block if condition is true
else: Executes this block if condition is false
```

if elif else statment

```
if (condition): statement
elif (condition):statement
else: statement
```

try except block

```
the code will go in the try block first if it's generat an error it will go to the except bloc
if not the try block will be executed and the except block will be skipped
try:
    statment
except:
    statment
```

[ ]:
```
#if statement
i = 10
if (i > 15):
       print ("10 is less than 15")
print ("I am Not in if")
```

[ ]:
```python
# Short Hand if statement
i = 10
if i < 15: print("i is less than 15")
```

[ ]:
```python
# Short Hand if-else statement
i = 10
print(True) if i < 15 else print(False)
```

[ ]:
```python
#if else statment
i = 20;
if (i < 15):
    print ("i is smaller than 15")
    print ("i'm in if Block")
else:
    print ("i is greater than 15")
    print ("i'm in else Block")
print ("i'm not in if and not in else Block")
```

[ ]:
```python
#if elif else statment
i = 20
if (i == 10):
    print ("i is 10")
elif (i == 15):
    print ("i is 15")
elif (i == 20):
    print ("i is 20")
else:
    print ("i is not present")
```

[ ]:
```python
astr = 'Hello Bob'
try:
    istr = int(astr)
except:
    istr = -1
```

[ ]:
```python
astr = '123'
try:
    istr = int(astr)
except:
    istr = -1

print ( istr)
```

1.1 Loops

1.1.1 For

For in loops

```
for var in iterable:
     statements
```

For range() function

```
for i in range(start, end , step):
    statments
```

For enumerate() function ##

```
for key , value in enumrate:( ):
    statments
```

For zip() function ## is used to combine 2 similar containers(list-list or dict-dict)

```
for i, j in zip( list 1, list 2):
        statments
```

1.1.2 While

while loop

```
while expression:
    statement(s)
```

[ ]:
```
# Iterating over a list
print("List Iteration")
l = ["hello", "world", "!"]
for i in l:
    print(i)
```

[ ]:
```
# Iterating over a String
print("\nString Iteration")
s = "hello"
for i in s :
    print(i)
```

[ ]:
```
# Iterating over dictionary
print("\nDictionary Iteration")
d = dict()
d['xyz'] = 123
d['abc'] = 345
for i in d :
    print(i , d[i])
```

[ ]:
```python
# printing a number
for i in range(10):
    print(i, end=" ")
print()
```

[ ]:
```python
# using range for iteration
l = [10, 20, 30, 40]
for i in range(len(l)):
    print(l[i], end=" ")
print()
```

[ ]:
```python
# performing sum of first 10 numbers
sum = 0
for i in range(1, 10):
    sum = sum + i
print("Sum of first 10 numbers :", sum)
```

[ ]:
```python
for key, value in enumerate(['The', 'Big', 'Bang', 'Theory']):
    print(key, value)
```

[ ]:
```python
# initializing list
questions = ['name', 'colour', 'shape']
answers = ['apple', 'red', 'a circle']

# using zip() to combine two containers
# and print values
for question, answer in zip(questions, answers):
    print('What is your {0}?  I am {1}.'.format(question, answer))
```

[ ]:
```python
# Single statement while block
count = 0
while (count < 5): count += 1; print("Hello world")
```

[ ]:
```python
a = [1, 2, 3, 4]
while a:
    print(a.pop())
```

2 Statments

2.0.1 Break

break

2.0.2 Continue

continue

2.0.3 pass

pass

[10]:
```
# demonstrate break statement

s = 'welcomehome'
# Using for loop
for letter in s:

    print(letter)
    # break the loop as soon it sees 'e'
    # or 's'
    if letter == 'e' or letter == 's':
        break

print("Out of for loop")
```

```
g
e
Out of for loop
```

[ ]:
```
# loop from 1 to 10
for i in range(1, 11):

    # If i is equals to 6,
    # continue to next iteration
    # without printing
    if i == 6:
        continue
    else:
        # otherwise print the value
        # of i
        print(i, end = " ")
```

[ ]:
```
li =['a', 'b', 'c', 'd']

for i in li:
    if(i =='a'):
        pass
    else:
        print(i)
```

3 String

3.1 Creating a String

with single Quotes

```
String1 = 'Welcome to the  World'
```

with double Quotes

```
String1 = "I'm a dev"
```

with triple Quotes

```
String1 = '''I'm a dev and I live in a world of "developers"'''
```

Quotes allows multiple lines

```
String1 = '''Geeks
            For
            Life'''
```

3.2 Accessing characters

a	f	w	g	t	y	u	i
0	1	2	3	4	5	6	7
-8	-7	-6	-5	-4	-3	-2	-1

String[index]

3.3 String Slicing

String[start:end:step]

3.4 String common functions

OPERATOR	DESCRIPTION	SYNTAX
s.startswith(prefix,start,end)	Returns True if a string starts with the given prefix otherwise returns False	s.startswith('an')
s.endswith(suffix,start,end)	Returns True if a string ends with the given suffix otherwise returns False	s.endswith('on')
s.strip(chars)	It return a copy of the string with both leading and trailing characters removed	s.strip(' low')

OPERATOR	DESCRIPTION	SYNTAX
s.count(sub,start,end)	Return the number of (non-overlapping) occurrences of substring sub in string	s.count("he")
s.partition(sep)→ (before,sep,after)	splits the string at the first occurrence of the separator and returns a tuple.	s.partition('is')
s.index(chr,start,end])	Returns the position of the first occurrence of substring in a string	pos = ch.index(ch1,2)
s.find(sub[,start[,end]])	Return the lowest index of a string.	s.find('hi')
s.is…()	tests on chars categories	s.isalpha()
s.upper()	lower case letters converted to upper case.	s.upper()
s.lower()	upper case letters converted to lower case.	s.lower()
s.swapcase()	Converts lower case letters to upper case and vice versa.	
s.swapcase()		
s.casefold()	Returns the string in lowercase which can be used for caseless comparisons.	s.casefold()
s.capitalize()	Return a word with its first character capitalized.	s.capitalize()
s.encode(encoding)	Encodes the string into any encoding supported by Python.Default encoding is utf-8.	
s.split([sep])	Return a list of the words of the string,If the optional second argument sep is absent or None	s.split()
s.join(seq)	Concatenate a list or tuple of words with intervening occurrences of sep.	s.join(list1)

[ ]:
```python
# characters of String

String1 = "helloworld"
print(String1[0])
```

[ ]:
```python
# you can't add string to int/float
print("hello"+2) # error
```

[ ]:
```
# but can be transformed to str
print("hello"+str(2))
```

[ ]:
```
#string repetation
for i in range(5):
print("hello world")
```

[ ]:
```
print("mohamed "*5)
```

[ ]:
```
# string slicing
sentence = "hello world"
for i in range(len(sentence)):
print(i ,end= " ")
print()
for char in sentence:
print(char ,end= " ")
```

[11]:
```
# demonstrate String slicing

# Creating a String
String1 = "welcometopython"
print(String1[3:12])
```

```
cometopyt
```

[ ]:
```
# .endswith() function

text = "welcome for geeks."

# returns False
result = text.endswith('for geeks')
print (result)
```

[ ]:
```
# .startsswith() function

text = "geeks for ever."

# returns False
result = text.startswith('for geeks')
print (result)
```

[ ]:
```
string = """    hello my dear    """

# prints the string without stripping
print(string)

# prints the string by removing leading and trailing whitespaces
```

```
print(string.strip())

# prints the string by removing geeks
print(string.strip(' dear'))
```

[ ]:
```
# count() method without optional parameters

# string in which occurrence will be checked
string = "eye for eye"

# counts the number of times substring occurs in
# the given string and returns an integer
print(string.count("geeks"))
```

[ ]:
```
string = "pawan is a good"

# 'is' separator is found
print(string.partition('is '))

# 'not' separator is not found
print(string.partition('bad '))

string = "pawan is a good, isn't it"

# splits at first occurrence of 'is'
print(string.partition('is'))
```

[ ]:
```
# initializing target string
ch = "geeksforgeeks"

# initializing argument string
ch1 = "geeks"

# using index() to find position of "geeks"
# starting from 2nd index
# prints 8
pos = ch.index(ch1,2)

print ("The first position of geeks after 2nd index : ",end="")
print (pos)
```

[ ]:
```
#find word
word = 'geeks for geeks'

# returns first occurrence of Substring
result = word.find('geeks')
print ("Substring 'geeks' found at index:", result )
```

[ ]:
```
# working of upper() function
text = 'geeKs For geEkS'

print(text)
# upper() function to convert
# string to upper_case
print(text.upper())
```

[ ]:
```
# working of upper() function
text = 'geeKs For geEkS'

print(text)
# upper() function to convert
# string to upper_case
print(text.lower())
```

[ ]:
```
# swapcase() method

string = "gEEksFORgeeks"

# prints after swappong all cases
print(string.swapcase())

string = "striver"
print(string.swapcase())
```

[ ]:
```
# Python program to convert string in lower case
string =" GEEKSFORGEEKS"

# print lowercase string
print(" lowercase string: ",string.casefold())
```

[ ]:
```
# capitalize() first letter of
# string.
name = "geeks for geeks"

print(name.capitalize())
```

[ ]:
```
import base64

# all encodings available

from encodings.aliases import aliases

# Printing list available
print("The available encodings are : ")
print(aliases.keys())
```

```
# Python code to demonstrate
# encode()

# initializing string
str = "geeksforgeeks"

#printing the encoded string
print ("The encoded string in base64 format is : ", )
print( base64.b64encode(str.encode('ascii')))
```

[ ]:
```
word = 'geeks:for:geeks'

# Splitting at ':'
print(word.split(':'))

word = 'CatBatSatFatOr'

# Splitting at 3
print([word[i:i+3] for i in range(0, len(word), 3)])
```

[ ]:
```
# list join
list1 = ['1','2','3','4']

s = "-"

# joins elements of list1 by '-'
print( s.join(list1) )
```