Day 8/8/21

IDLE = Editor + Interpreter + runtime environment
IDLE – (Integrated
IdlE Features :- • Syntax autocompletion
• Inbuild python modules are available for your programming.

pip uninstall package name (for uninstalling)
pip install package name (for installing)

ex:1) name = "My Name is yogita"
print (name.lower())
Output:- my name is yogita
ex: 2) name = " Snehal "
print (name.upper())
output:- SNEHAL

ex:3) name = "LALALALALALALALALALALAA"
print (name.lower())
Output:- lalalalalalalalalalaa

ex:4) name = """this string containing whitespaces ~~print (name.string())~~ but why i dont know so that i am removing it using strip() method"""
print (name.strip())
output:-

ex 5) paragraph = "hey how are you, are you doing a wrong code, oops you must be replaced by using replace() method"
print (paragraph.replace ("h", "0"))

Day 6
7/8/22

IDLE = Editor + Interpreter + runtime environment
IDLE - (Integrated
IdlE features :- • Syntax autocompletion
• Inbuild python modules are available for your programming.

pip uninstall package name (for uninstalling)
pip install package name (for installing)

ex:1) name = "My Name is yogita"
print (name.lower())
Output:- my name is yogita
ex: 2) name = " Snehal "
print (name.upper())
output:- SNEHAL

ex:3) name = "LALALALALALALALALALALALAA"
print (name.lower())
output:- lalalalalalalalalalaa

ex:4) name = "" this string containing whitespaces ~~print (name.strip())~~ but why i dont know so that i am removing it using strip() method""
print (name.strip())
output:-

ex 5) paragraph = "hey how are you, are you doing a wrong code, oops you must be replaced by using replace() method"
print (paragraph.replace ("h", "0"))

ex:6

```
team = "hello, team, pranav, gourav, snehal,
        vaishnavi, pratiksha, ganesh, we must
        split by any operator"
print (type(team))
print (team.split(", "))
print (type(team))
```

Output:- ['hello', 'team', 'pranav', 'gourav', 'snehal', 'vaishnavi', 'pratiksha', 'ganesh', 'we must split by any operator']
<class 'str'>

Concatenation of String

1) Str1 = "Python"
   Str2 = "Internship"

```
result = str1 + str2
print (result)
```

output:- PythonInternship

2) Space = " "
   result = Str1 + ~~str2~~ space + Str2
   Print (result)

output:- python Internship

String format method

1) name = "My name is yogita, and I am {} years old"

```
Print (name.format (19))
```

Oulput:- My name is yogita, and I am 19 years old

2) name = "My name is yogita, and I am {} years old, I like {}, My collage is {}"

```
print (name.format (17, "chocolates", "GPP"))
```

My name is yogita, and I am 19 years old, I like chocolates, My collage is GPP

```
print(bool())                          output:- False
print(bool("is this true"))                   = True
print(bool(1234))                             = True
```

- Any string is always True,
- Any Number is always True
- Empty strings are false, 0 is false
- blank is always false

- print(bool(0)) - False
- print(bool("")) - False

- print(bool(())) - False
- print(bool([])) False
- print(bool({})) False
- print(bool(false)) false
- print(bool(none)) false

Operators :

Operators are special symbols that carry out arithmetic operations such as :-
- used to assign values to variables

→ Arithmetic operators (+, -, *, /, %, ** (exponentiation), // (floor division))
→ Assignment operators (=)
→ Comparison operators (==, !=, >, <, >=, <=)
→ Logical operators (and, or, not)
→ Identity operators (is, is not)
→ Membership operators (in, not in)
→ Bitwise operators (|, &, ~, ^, >>, <<)

→ Shorthand operators (+=, -=, *=, /=, %=)
→ and (true & true = true)
→ or (If any one is true then condition is true)
→ not (Invert the result True → false
[True ⇌ False] False → True

operators

*) Assign Arithmetic operators

1) Prg. (+) Addition operator

```
x = 10
y = 20
print ('x+y =', x+y)
```

output:- x + y = 30

2) (-) Substraction operator

```
x = 10
y = 20
print ('x-y =', x-y)
```

⇒ x - y = -10

3) (*) Multiplication operator

```
x = 10
y = 20
print ('x*y =', x*y)
```

⇒ x * y = 200

4) (/) Division operator

```
x = 10
y = 20
print ('x/y =', x/y)
```

⇒ x/y = 0.5

5) (%) modulo operator)

```
x = 20
y = 10
print ('x % y =', x % y)
```

⇒ x % y = 0

6) (//) floor division)

```
x = 50
y = 20
print ('x // y =', x//y)
```

⇒ x // y = 2

7) (**) Exponent operator

```
x = 10
y = 4
print ('x ** y =', x ** y)
```

⇒ x ** y = 10,000

~~8) Assignment operator (=)~~

* Comparison operators

1) (==) Equal comparison op.

```
x = 20
y = 25
print ('x==y is', x==y)
```

⇒ x==y is false

2) (!=) Not equal op.

```
x = 20
y = 25
print ('x!=y is', x!=y)
```

⇒ x!=y is true

3) (>) Greater than op.

```
x=20
y=25
print('x>y is', x>y)
```

⇒ x>y is False

4) (<) less than op.

```
x=20
y=25
print ('x<y is', x<y)
```

⇒ x<y is True.

5) (>=) Greater than or equal op.

```
x = 20
y = 25
print('x >= y is', x >= y)
```

⇒ x >= y is False

6) (<=) less than or equal to op.

```
x=20
y=25
print('x <= y is', x <= y)
```

⇒ x <= y is True

* Assignment operators

ex:- 5 = 2 + 3.

1) (+=) Add AND op.

```
x = 10
x += 5
Print(x)
```

⇒ x = 15.

2) (-=) Subtract AND op.

```
x = 10
y = 7
Print(x)
```

⇒ x = 3

3) (*=) Multiply AND op.

```
x = 10
x *= 5
Print (x)
```
=> x = 50

4) (/=) Divide AND op.

```
x = 10
x /= 5
Print (x)
```
=> x = 2

5) (%=) Modulus AND op.

```
x = 10
x %= 5
Print (x)
```
=> x = 0

6) (**=) Exponent AND op.

```
x = 10
x **= 5
print (x)
```
=> x = 1,00,000

7) (//=) Floor division

```
x = 7
x //= 4
print (x)
```

=> x = 1

* Logical operators

1) Logical AND op.

```
x = 10
y = 20
print('x and y is', x and y)
```

⇒ x and y is True.

2) Logical OR op.

```
x = 10
y = 20
print('x or y is', x or y)
```

⇒ x or y is True

3) Logical NOT op.

```
x = 10
y = 20
print('Not x is', not x)
```

⇒ not x is false

* ~~Identity operators~~

* Bitwise op.

1) (&) Binary AND op.

```
a = 10
b = 4
print (a & b)
```

⇒ 0

2) (|) Binary OR Bitwise op.

```
a = 10
b = 4
print (a | b)
```

⇒ 14

3) (^) Binary XOR Bitwise op.

```
a = 10
b = 4
print (a ^ b)
```

→ 14

4) (~) Binary ones complement bitwise op.

```
a = 10
b = 4
print (~a)
```

⇒ -11

5) (<<) Binary left shift bitwise op.

```
a = 10
b = 4
print (a<<2)
```

⇒ 40

6) (>>) Binary right shift bitwise op.

```
a = 10
b = 4
print (a>>2)
```

⇒ 2

ex: * Identity operators

1) x = ["grapes", "banana"]
   y = ["grapes", "banana"]
   new = x # Are x & new same ???
   print (x is new)

output:- True

2) # Are x and new different ???
   print (x is not new)

output:- False

print (x != y) ⇒ False

* Membership operators

1) x = 'Hello world'
   y = {1: 'a', 2: 'b'}

   print ('H' in x)

   print ('hello' not in x)

Output:- True
True

list []

Lists are used to store multiple items in a single variable.

ex:- print fruitlist = ["grapes", "banana", "apples"]

output:- ['grapes', 'banana', 'apples']

2) print(len(fruitlist))

⇒ 3

3) numlist = [1, 5, 10, 9, 3]
booleanlist = [True, False, False]
mixDatatype list = ["shiv", 25, True, 500000, "male"]

print (numlist)
print (booleanlist)
print (mixDatatypelist)

⇒ [1, 5, 10, 9, 3]
mixDatatype list = ["shiv", 25, True, 50000, "male"]

Assignment.py

~~Assignment~~ string.methods()

~~name.capitalize(~~
~~print~~

1) capitalize() -
→ converts the first character to upper case

Program -

```
name = "my name is yogita"
print (name.capitalize())
```

Output :- My name is yogita

2) casefold() -
converts string into lower case. ~~ee~~

~~prog~~ - ~~My nar~~ name = "MY name is yogita."
print (name.casefold())

output :- my name is yogita

3) center() - Returns a centered string.

```
name = "my name is yogita"
prog - x = name.center(50)
       print(x)
```

output :- my name is yogita

4) count() - Returns the number of times a specified value occurs in a string

Prog:- name = "my name is yogita"
print (name.count("yogita"))

Output:- 1

5) encode() - Returns an encoded version of the string.

Program:- name = "my name is yogita"
print (name.encode())

output:- b' my name is yogita'

6) endswith() - Returns true if the string ends with the specified value.

Prog:- name = "my name is yogita"
print (name.endswith("yogita"))

Output:- True

7) expandtabs() - sets the tab size of the String.

Prog:- name = "my\t name is \t yogita"
result = name.expandtabs(2)
print (result)

output:-

my name is yogita

8) find() - Searches the string for a specified value & returns the position of where it was found.

prog :- name = "my name is yogita"
print (name.find(" is "))

output:- 8

9) format() - formats the specified value(s) & insert them inside the string's & returns the formatted string.

Program:-
txt = "for only {price:.2f} dollars!"
print(txt.format(price=49))

output :-
For only 49.00 dollars!