# ses 1 presentation intro

October 8, 2021

## 0.1 Taking an Input

[2]:
```
# Python program showing
# a use of input()

num =int( input ("Enter number :") )
print(num)
name1 = (input("Enter name : "))
print(name1)
```

```
Enter number :1
1
Enter name : ad
ad
################################################################################
########################

type of number <class 'int'>
type of name <class 'str'>
```

[ ]:
```
# Printing type of input value
print ("type of number", type(num))
print ("type of name", type(name1))
```

## 0.2 Variable Names

| | | |
|---|---|---|
| 1. | Must start with a letter (or underscore). | |
| 2. | Can include letters, digits, and underscores, but nothing else. | |
| | Example: | Case matters: age = 11 |
| | Valid: | _moo_cow, cep3, I_LIKE_TRASH |
| | Invalid: | 49ers, @home |
| | Different: | spam , Spam , SPAM |
| | Reserved Words | and,del,for,is,raise,assert,elif,from,lambda,return,break,<br>if,or,while,continue,exec,import,pass,yield,def,finally,in,p |

| 3. | Python convention: pothole_case |
|---|---|

[14]:
```
_moocow = 1
print("_moocow:",_moocow)
```

```
_moocow: 1
```

[ ]:
```
cep3 = 2
print("cep3:",cep3)
```

[ ]:
```
I_LIKE_TRASH = 3
print("I_LIKE_TRASH:",I_LIKE_TRASH)
```

[ ]:
```
#not a valid named variable
print('not a valid named variable \n')
```

[ ]:
```
print("triple quates:","""49ers = 4
print(49ers)
@home = 5
print(@home)
""")
```

### 0.3 Arithmetic operators:

| OPERATOR | DESCRIPTION | SYNTAX |
|---|---|---|
| + | Addition: adds two operands | x + y |
| - | Subtraction: subtracts two operands | x - y |
| * | Multiplication: multiplies two operands | x * y |
| / | Division (float): divides the first operand by the second | x / y |
| // | Division (floor): divides the first operand by the second | x // y |
| % | Modulus: returns the remainder when first operand is divided by the second | x % y |
| ** | Power : Returns first raised to power second | x ** y |

```
[ ]: # Examples of Arithmetic Operator
a = 9
b = 4
```

```
[ ]: # Addition of numbers
add = a + b
print("add:",add)
```

```
[ ]: # Subtraction of numbers
sub = a - b
print("sub:",sub)
```

```
[ ]: # Multiplication of number
mul = a * b
print("mul:",mul)
```

```
[ ]: # Division(float) of number
div1 = a / b
print("div1:",div1)
```

```
[ ]: # Division(floor) of number
div2 = a // b
print("div2:",div2)
```

```
[ ]: # Modulo of both number
mod = a % b
print("mod:",mod)
```

```
[ ]: # Power
po = a ** b
print("po:",po)
```

### 0.4 Relational Operators:

| OPERATOR | DESCRIPTION | SYNTAX |
|---|---|---|
| > | RelationalGreater than: True if left operand is greater than the right | x > y |
| < | Less than: True if left operand is less than the right | x < y |
| == | Equal to: True if both operands are equal | x == y |
| != | Not equal to - True if operands are not equal | x != y |

| OPERATOR | DESCRIPTION | SYNTAX |
|---|---|---|
| >= | Greater than or equal to: True if left operand is greater than or equal to the right | x >= y |
| <= | Less than or equal to: True if left operand is less than or equal to the right | x <= y |

[ ]:
```
# Examples of Relational Operators
a = 13
b = 33
```

[ ]:
```
# a > b is False
print("a > b :",a > b)
```

[ ]:
```
# a < b is True
print("a < b:",a < b)
```

[ ]:
```
# a == b is False
print("a == b:",a == b)
```

[ ]:
```
# a != b is True
print("a != b:",a != b)
```

[ ]:
```
# a >= b is False
print("a >= b:",a >= b)
```

[ ]:
```
# a <= b is True
print("a <= b:",a <= b)
```

### 0.5 Logical operators:

| OPERATOR | DESCRIPTION | SYNTAX |
|---|---|---|
| and | Logical AND: True if both the operands are true | x and y |
| or | Logical OR: True if either of the operands is true | x or y |
| not | Logical NOT: True if operand is false | not x |

| OPERATOR | DESCRIPTION | SYNTAX |
|---|---|---|
| any | Returns true if any of the items is True | any([False, True, False, False]) |
| all | Returns true if all of the items are True | all([False, True, True, False]) |

[ ]:
```
# Examples of Logical Operator
a = True
b = False
```

[ ]:
```
# Print a and b is False
print("a and b:",a and b)
```

[ ]:
```
# Print a or b is True
print("a or b:",a or b)
```

[ ]:
```
# Print not a is False
print("not a:",not a)
```

[ ]:
```
# Here the method will short-circuit at the
# second item (True) and will return True.
print ("any:",any([False, True, False, False]))
```

[ ]:
```
# Here the method will short-circuit at the
# first item (False) and will return False.
print ("all:",all([False, True, True, False]))
```

### 0.6 Bitwise operators:

| OPERATOR | DESCRIPTION | SYNTAX |
|---|---|---|
| & | Bit wise AND | x & y |
| | Bit wise OR | x y |
| ~ | Bit wise NOT | x ~ y |
| ^ | Bit wise XOR | x ^ y |
| >> | Bit wise Right Shift | x >> y |
| << | Bit wise Left Shift | x << y |

[ ]:
```
# Examples of Bitwise operators
a = 10
b = 4
```

[ ]:
```
# Print bitwise AND operation
print("bitwise AND operation a & b:",a & b)
```

[ ]:
```
# Print bitwise OR operation
print("bitwise OR operation a | b:",a | b)
```

[ ]:
```
# Print bitwise NOT operation
print("bitwise NOT operation a:",~a)
```

[ ]:
```
# print bitwise XOR operation
print("XOR operation a ^ b:",a ^ b)
```

[ ]:
```
# print bitwise right shift operation
print("right shift operation a >> 2 :",a >> 2)
```

[ ]:
```
# print bitwise left shift operation
print("left shift operation a << 2:",a << 2)
```

### 0.7 Assign Variables

| OPERATOR | DESCRIPTION | SYNTAX |
|---|---|---|
| = | Assign value of right side of expression to left side operand | x = y + z |
| , , | Assign multi values of right side of expression to left side operands | x,y,e,q = 5,2,9,1 |
| += | Add AND: Add right side operand with left side operand and then assign to left operand | a+=b a=a+b |

And the same goes fo similar operators

[15]:
```
a = 5
print("a:",a)
```

```
a: 5
```

[ ]:
```
x,y,e,q = 5,2,9,1
print("x,y,e,q:",x,y,e,q)
print("x:",x)
print("y:",y)
print("e:",e)
print("q:",q)
```

[ ]:
```
a+=x
print("a+=x:",a)
```

## 0.8 Print

| Print in specific position | | |
|---|---|---|
| print(f”) | print variable in specific place | print(f’hello world from {x} , have a good day’) |
| ptint(”.format()) | print variable in specific place | print(‘hello world from {} , have a good day’.format(x)) |
| print(’ %s , %d , %b’ %(x , y, z)) | print variable in specific place (old formting) | print(‘hello world from %s , have a good day’ %x) |

[16]:
```
x= input ('please enter your name : \n')
print(f'hello world from {x} , have a goof day')
```

```
please enter your name :
as
hello world from as , have a goof day
```

[ ]:
```
y = x
print('hello world from {} , have a good day {}'.format(x*2,y))
```

[ ]:
```
print("%s=%s" % ("pi", 3.14159))
```