Python

PCEP-PREP DAY 1

Estevan A.

Subjects	Goals
Perform Operations using Data Types and Operators	• Assign data types to variables • Perform data and data type operations • Perform Arithmetic, Comparison and Logical Operations • Determine the sequence of execution based on operator precedence • Select the appropriate operator to achieve the intended result
Control Flow with Decisions and Loops	• If - else statements • elif statements • while and for loops • break; continue; pass
Exercises/Review	• complete exercises given in class

6 STANDARD DATA TYPES

- Numbers
- String
- List
- Tuple
- Dictionary
- Boolean

Numbers

Four Numerical Types

- int (signed integers)
- long (long integers, can be octal or hex)
- float (floating point real values)
- complex (complex numbers)

int	long	float	complex
10	51924361L	0.0	3.14j
100	-0x19323L	15.20	45.j
-786	0122L	-21.9	9.322e-36j
080	0xDEFABCECBDAECBFBAEI	32.3+e18	.876j
-0490	535633629843L	-90.	-.6545+0J
-0x260	-052318172735L	-32.54e100	3e+26J
0x69	-4721885298529L	70.2-E12	4.53e-7j

Assignment

counter = 100

miles = 1000.0

size = 1.34234

String

Strings in Python are identified as a contiguous set of characters represented in quotation marks

```
str = 'Hello World!'

print str          # Prints complete string
print str[0]       # Prints first character of the string
print str[2:5]     # Prints characters starting from 3rd to 5th
print str[2:]      # Prints string starting from 3rd character
print str * 2      # Prints string two times
print str + "TEST" # Prints concatenated string
```

OUTPUT

```
Hello World!
H
llo
llo World!
Hello World!Hello World!
Hello World!TEST
```

List

A list contains items separate by commas and enclosed within square brackets [ ].

```
list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john']

print list          # Prints complete list
print list[0]       # Prints first element of the list
print list[1:3]     # Prints elements starting from 2nd till 3rd
print list[2:]      # Prints elements starting from 3rd element
print tinylist * 2  # Prints list two times
print list + tinylist # Prints concatenated lists
```

OUTPUT

```
['abcd', 786, 2.23, 'john', 70.2]
abcd
[786, 2.23]
[2.23, 'john', 70.2]
[123, 'john', 123, 'john']
['abcd', 786, 2.23, 'john', 70.2, 123, 'john']
```

Tuples

A tuple is another sequence data type like a list but is enclosed with ( ) instead. Key difference between List vs Tuples is that list are able to change size and elements where tuples cannot. It can be seen as read-only.

```
tuple = ( 'abcd', 786 , 2.23, 'john', 70.2  )
tinytuple = (123, 'john')

print tuple               # Prints the complete tuple
print tuple[0]            # Prints first element of the tuple
print tuple[1:3]          # Prints elements of the tuple starting from 2nd till 3rd
print tuple[2:]           # Prints elements of the tuple starting from 3rd element
print tinytuple * 2       # Prints the contents of the tuple twice
print tuple + tinytuple   # Prints concatenated tuples
```

OUTPUT

```
('abcd', 786, 2.23, 'john', 70.2)
abcd
(786, 2.23)
(2.23, 'john', 70.2)
(123, 'john', 123, 'john')
('abcd', 786, 2.23, 'john', 70.2, 123, 'john')
```

Dictionary

```
dict = {}
dict['one'] = "This is one"
dict[2]     = "This is two"

tinydict = {'name': 'john','code':6734, 'dept': 'sales'}


print dict['one']       # Prints value for 'one' key
print dict[2]           # Prints value for 2 key
print tinydict          # Prints complete dictionary
print tinydict.keys()   # Prints all the keys
print tinydict.values() # Prints all the values
```

Dictionary are kind of hast table type where it consist of key-value pairs. It can be almost any python type but usually numbers or strings. Dictionaries are enclosed with `{}` and values can be assigned or accessed with `[]`.

OUTPUT

```
This is one
This is two
{'dept': 'sales', 'code': 6734, 'name': 'john'}
['dept', 'code', 'name']
['sales', 6734, 'john']
```

Boolean

```python
a = True
# display the value of a
print(a)

# display the data type of a
print(type(a))
```

Boolean type is one of built-in data types which represents one of the two values **True** or **False.**

Exercise 1

Module 1

1. Declare a variable named 'counter' and set to 100
2. Declare a variable named 'student' and set to your name
3. Declare a variable of type list named 'numbers' with at least 10 different integers

Exercise 2

Module 1

1. Assign a new variable 'copy' with counter variable
2. Print only the first 3 letters of 'student'
3. Print the second half of the list 'numbers'
4. Assign new variables to the results of 1 , 2 and 3.

Exercise 3

Module 1

1. Declare a tuple named 'class' with 2 integers and 2 strings
2. Declare a dictionary named 'book' with keys as integers, and values as books names

Arithmetic Operators

Operator	Name	Example
+	Addition	10 + 20 = 30
-	Subtraction	20 – 10 = 10
*	Multiplication	10 * 20 = 200
/	Division	20 / 10 = 2
%	Modulus	22 % 10 = 2
**	Exponent	4**2 = 16
//	Floor Division	9//2 = 4

Arithmetic Operators are used to perofrm mathematical operations on numerical values. These include:

- addition
- subtraction
- multiplication
- division
- modulus
- exponent
- floor division

Comparison Operators

Operator	Name	Example
==	Equal	4 == 5 is not true.
!=	Not Equal	4 != 5 is true.
>	Greater Than	4 > 5 is not true.
<	Less Than	4 < 5 is true.
>=	Greater than or Equal to	4 >= 5 is not true.
<=	Less than or Equal to	4 <= 5 is true.

Comparison operators compare the values on either sides of them and decide the relation among them. They are also called relational operators. These operators are equal, not equal, greater than, less than, greater than or equal to and less than or equal to.

Assignment Operators

Operator	Name	Example
=	Assignment Operator	a = 10
+=	Addition Assignment	a += 5 (Same as a = a + 5)
-=	Subtraction Assignment	a -= 5 (Same as a = a - 5)
*=	Multiplication Assignment	a *= 5 (Same as a = a * 5)
/=	Division Assignment	a /= 5 (Same as a = a / 5)
%=	Remainder Assignment	a %= 5 (Same as a = a % 5)
**=	Exponent Assignment	a **= 2 (Same as a = a ** 2)
//=	Floor Division Assignment	a //= 3 (Same as a = a // 3)

Assignment operators are used to assign values to variables. These operators include simple assignment operator, addition assign, subtraction assign, multiplication assign, division and assign operators etc.

Exercise 4

Module 1

x = 6

y = 7

Calculate the following:

1. x + x + y
2. x * y + x
3. x ** x
4. y // x

Exercise 5

Module 1

x = 8

y = 6

Calculate the following:

1. x - x + y
2. x ** 2 + y
3. x ** x ** y ** 2
4. x % y

Exercise 6

Module 1

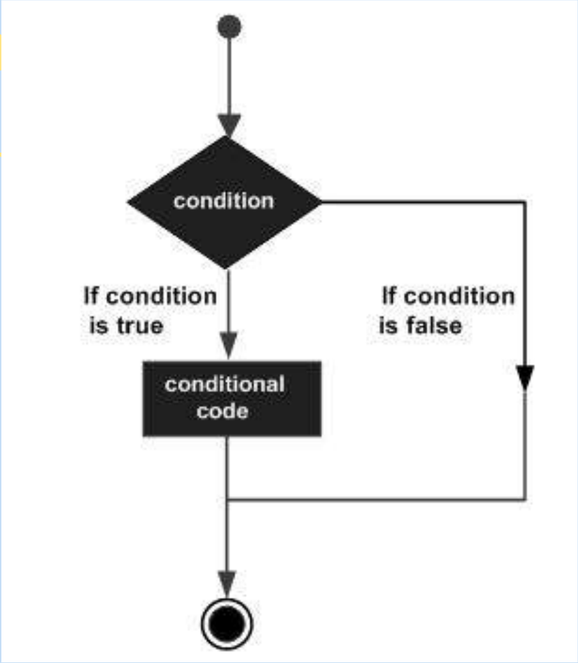
x = 5

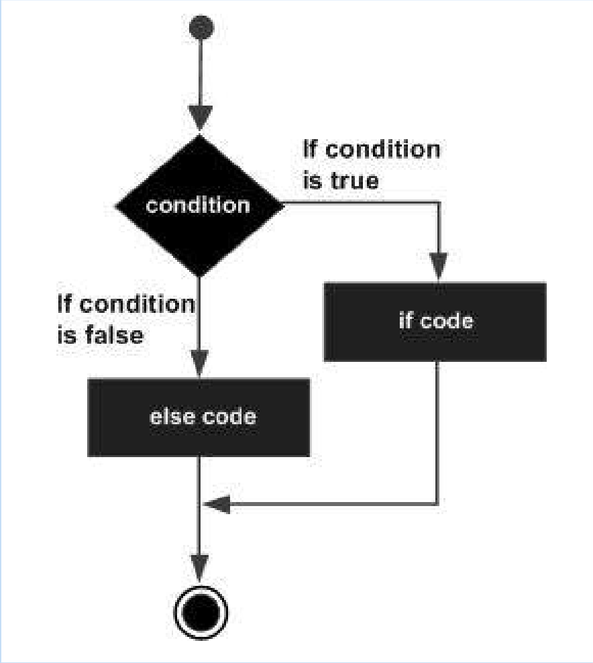
y = 11

Calculate the following boolean values:

1. x**2 >= y
2. x ** 3 == y*10 + x*4 - x
3. y // x < x
4. y**x % x

If - else statements





Decision making is anticipation of consitions occuring while exection of the program and specifying actions taken accorind to the conditions.

```
if expression:
   statement(s)
else:
   statement(s)
```

elif statements

```
if expression1:
   statement(s)
elif expression2:
   statement(s)
elif expression3:
   statement(s)
else:
   statement(s)
```

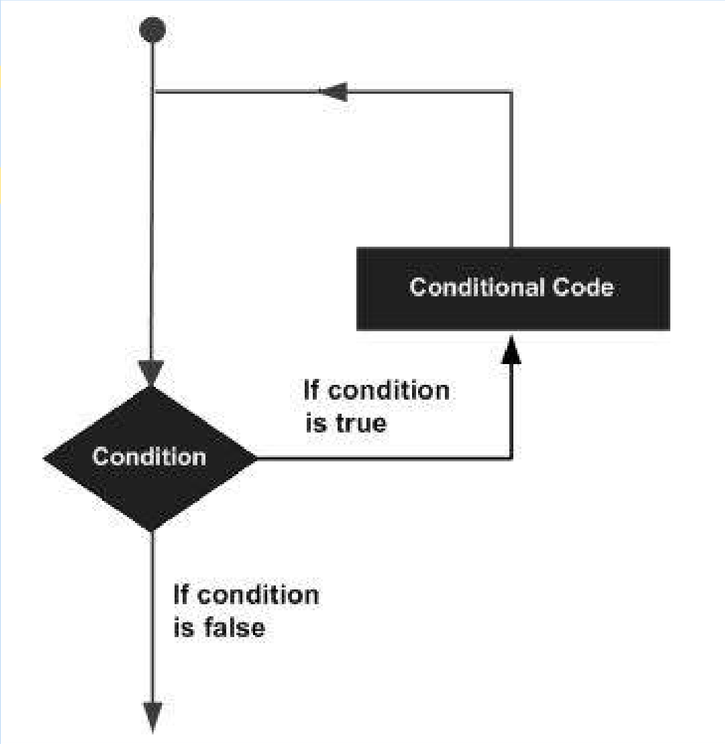
```
var = 100
if var == 200:
   print "1 - Got a true expression value"
   print var
elif var == 150:
   print "2 - Got a true expression value"
   print var
elif var == 100:
   print "3 - Got a true expression value"
   print var
else:
   print "4 - Got a false expression value"
   print var

print "Good bye!"
```

The elif statement allows you to check multiple expressions for TRUE and execute a block of code as soon as one of the conditions evaluates to TRUE.

Similar to the else, the elif statement is optional. However, unlike else, for which there can be at most one statement, there can be an arbitrary number of elif statements following an if.

Loops



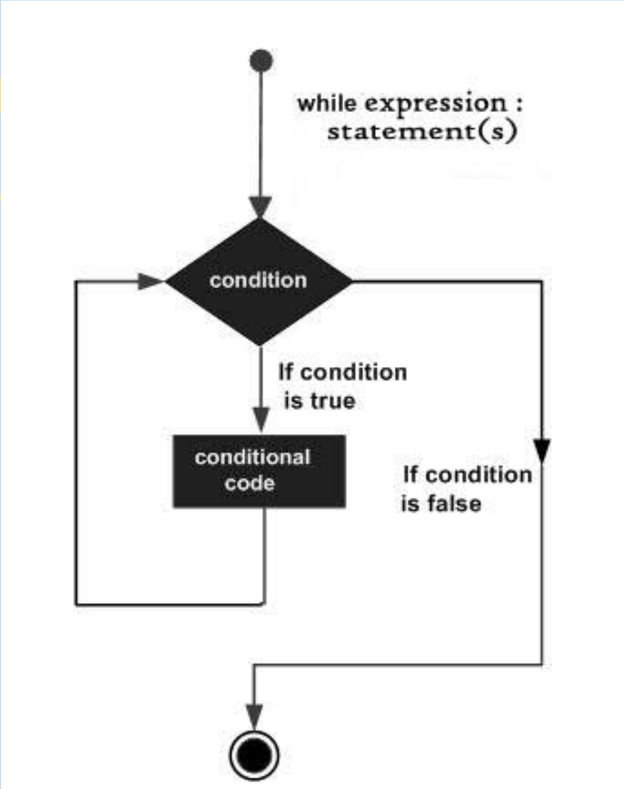
- While loop
- For loop

In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on. There may be a situation when you need to execute a block of code several number of times.

Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or group of statements multiple times. The following diagram illustrates a loop statement

While loop

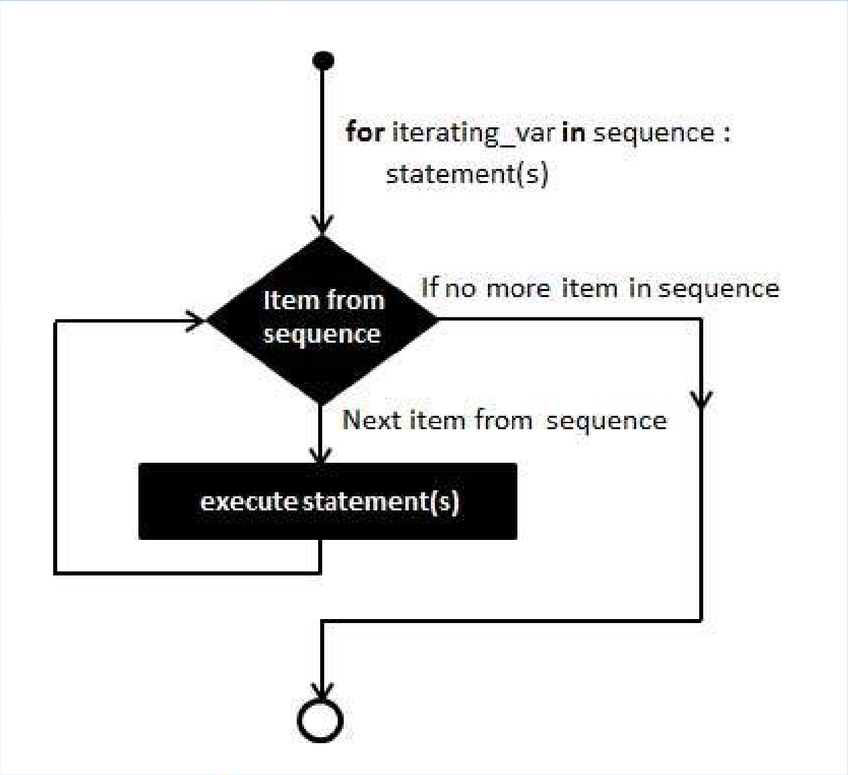


A while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.

```
count = 0
while (count < 9):
   print 'The count is:', count
   count = count + 1

print "Good bye!"
```

For loop



A for loop statement in Python programming language has the ability to iterate over the items of any sequence, such as a list or a string.

```
for letter in 'Python':     # First Example
   print 'Current Letter :', letter

fruits = ['banana', 'apple',  'mango']
for fruit in fruits:        # Second Example
   print 'Current fruit :', fruit

print "Good bye!"
```

CLASS EXERCISE

Topics

- Variables assignment
- Arithmetic operators
- Comparison operators
- Built in Functions
- Data Types
- Loops

M1Q1

What is the output of the below program?

```
a = -1
b = 1
a= ( a== b )

print ("Value of a is :", a, "value of b is :", b )
```

1. Value of a is True, value of b is : 1
2. Value of a is : False value of b is : 1
3. Error

M1Q2

Which variable gets the value of 5?

```
i = 11
j = i/2
k = int(i)/2
l = i/int(2)
m = int(i/2)

print ( j )
print ( k )
print ( l )
print ( m )
```

1. I
2. J
3. K
4. L
5. m

M1Q3

Consider the below code. Which of the statements will give error?

```
s1 = "a" + "b"    #Statement 1
s2 = "a" + 'c'    #Statement 2
s3 = "a"* "b"     #Statement 3
s4 = "a" / "b"    #Statement 4
```

1. Statement 1
2. Statement 2
3. Statement 3
4. Statement 4

Cont..

PYTHON

M1Q4

```
s1 = " Hello "
```

 # if len(s1.rstrip()) = 8 and len(s1.lstrip()) = 8 what is the total number of spaces in the string s1.

1. 2
2. 3
3. 6
4. 0

M1Q5

```
v=bool([False])
x=bool(3)
y=bool("")
z=bool(' ')
print("v is :", v)
print("X is :", x)
print("y is :", y )
print("Z is :", z)
```

Which of the above statements will print true as output?

1. x is : true
2. y is : true
3. v is : true
4. z is : true
5. x is : false
6. v is : false
7. y is : false
8. z is : false

M1Q6

```
a = -1
b = 1
a= ( a== b )

print("Value of a is :", a, "Value of b is :", b )
```

What is the output of the print statement?

1. Value of a is : -1, Value of b is : 1
2. Value of a is : 1, Value of b is : 1
3. Value of a is : 0, Value of b is : 1
4. Value of a is : False, Value of b is : 1
5. Value of a is : True, Value of b is : 1

PYTHON

Practice link

Click the link to join now.

quizizz