PDS_Wk_1-2

August 16, 2022

Control flow, determines how logic flows through a Python program (more simply, what is the sequence in which code statements would execute)?

[1]:
```
### To begin with, we look at the 'if, else' construct
x = 4
y = 4
if x > y:
    print('5 > 4')
elif x < y:
    print('5 < 4')
else:
    print('5 == 4')
```

```
5 == 4
```

[7]:
```
### The while statement (we start 'looping' here, the loop will continue while␣
 ↪the Condition is TRUE)
number = 23
running = True
while running:
 guess = int(input('Enter an integer : '))
 if guess == number:
        print('Congratulations, you guessed it.')
        # this causes the while loop to stop
        running = False
 elif guess < number:
        print('No, it is a little higher than that.')
 else:
        print('No, it is a little lower than that.')
else:
    print('The while loop is over.')
# Do anything else you want to do here
print('Done')
```

```
Enter an integer :  13

No, it is a little higher than that.

Enter an integer :  19

No, it is a little higher than that.
```

```
Enter an integer :  29

No, it is a little lower than that.

Enter an integer :  23

Congratulations, you guessed it.
The while loop is over.
Done
```

[17]:
```python
for x in range(0, 10): #the value of x will range from lower_limit to upper_limit-1
    print(x)
```

```
0
1
2
3
4
5
6
7
8
9
```

[23]:
```python
#print all values on the same line
for x in range(0, 10):
    print(x, end=" ")
print("End")
```

```
0 1 2 3 4 5 6 7 8 9 End
```

[24]:
```python
#Break statement to deliberately terminate a loop. This loop will terminate when the user enters '0'
while True:
    x = int(input('Enter a number : '))
    if x == 0:
        break
    print('The number is', x)
print('Done')
```

```
Enter a number :  3

The number is 3

Enter a number :  4

The number is 4

Enter a number :  9

The number is 9

Enter a number :  0
```

```
Done
```

[25]:
```
#continue hands control back to the loop, we will get back to checking if the␣
 ↪loop/terminal condition is TRUE or not
while True:
    x = int(input('Enter a positive number : '))
    if x == 0:
        break
    elif x < 0:
        print('The number must be positive')
        continue
    print('The number is', x)
print('Done')
```

```
Enter a positive number :  2
The number is 2
Enter a positive number :  -2
The number must be positive
Enter a positive number :  0
Done
```

[26]:
```
#Break when the user inputs a negative number
while True:
    x = int(input('Enter a positive number : '))
    if x == 0:
        continue
    elif x < 0:
        print('The number must be positive: Exiting')
        break
    print('The number is', x)
print('Done')
```

```
Enter a positive number :  4
The number is 4
Enter a positive number :  0
Enter a positive number :  8
The number is 8
Enter a positive number :  -2
The number must be positive: Exiting
Done
```

Introducing Functions: As you write large programs/programs as part of a large collaborative project, it is important to ***modularize*** your code than write one LARGE code file

[27]:
```
#Using python's built-in functions
x = int(input('Enter your assignment 1 mark: '))
y = int(input('Enter your assignment 2 mark: '))
z = int(input('Enter your assignment 3 mark: '))

xyz = [x, y, z] #array

if (x and y and z) in range(0, 34):
    print('Your highest mark is ' + str(max(xyz)))
    print('Your lowest mark is ' + str(min(xyz)))
    print('Your total mark is ' + str(sum(xyz)))
```

```
Enter your assignment 1 mark:  22
Enter your assignment 2 mark:  32
Enter your assignment 3 mark:  29

Your highest mark is 32
Your lowest mark is 22
Your total mark is 83
```

[ ]:
```
#Function call with *no* arguments/parameters
#define function
def get_grade():
     print('your grade is HD')
# end of function
get_grade()
```

```
your grade is HD
```

[33]:
```
def add(a,b):
   return(a+b)
print(add(5,4))
```

```
9
```

[35]:
```
# Function that takes in an integer, and returns a string
def get_mark(m):
    s= 'Your mark is '+str(m)
    return(s)

print(get_mark(99))
```

```
Your mark is 99
```

While using functions, you need to be mindful of the "scope" of a variable. Some variables are 'global' (everyone knows the US/Aussie PM) vs 'local' (everyone at home knows you and me, not necessarily outside of it)

[37]:
```
s = "Hi there!" #variable scope-from here to end of 'main' code
#define function
```

```
def get_mark(m):
    s = 'Your mark is ' + str(m)
    print('s1 = ' + s)
    return s
# end of function
get_mark(99)
print('s2 = ' + s)
```

```
s1 = Your mark is 99
s2 = Hi there!
```

[39]:
```
# Explicitly define a Global variable
s = "Hi there!"
#define function
print(s)
def get_mark(m):
    global s
    s = 'Your mark is ' + str(m) #s is being re-initialized/re-defined here
    print('s1 = ' + s)
    return s
# end of function
get_mark(99)
print('s2 = ' + s)
```

```
Hi there!
s1 = Your mark is 99
s2 = Your mark is 99
```

[40]:
```
#Assuming default/optional function arguments

#define function
def get_mark(m=100):
    s = 'Your mark is ' + str(m)
    return s
# end of function

print(get_mark()) #default value used
print(get_mark(77))
```

```
Your mark is 100
Your mark is 77
```

[42]:
```
#You don't need to pass arguments in the same order as the function definition␣
 ↪as long as you specify their values at the function call
def subtract(a, b):
    return a - b

c = subtract(9, 7)
```

```
print(c) #c = 2

d = subtract(a=9, b=7)
print(d) #d = 2

d = subtract(b=9, a=7)
print(d) #d = -2
```

```
2
2
-2
```

[43]:
```
#error on b
def subtract(a=1, b):
    return a - b
```

```
  Input In [43]
    def subtract(a=1, b):
                 ^
SyntaxError: non-default argument follows default argument
```

[46]:
```
#Acceptable
def subtract(a, b=1):
    return a - b

c = subtract(9, 7)
print(c) #c = 2

d = subtract(a=9, b=7)
print(d) #d = 2

e = subtract(b=5, a=7)
print(e) #e = 2

f = subtract(7)
print(f) #f = 6

g = subtract(1, b=7)
print(g) #g = -6
```

```
2
2
2
6
-6
```

[48]:
```
#keyword arguments

def subtract(a=1, b=1):
    return a - b

c = subtract(9, 7)
print(c) #c = 2

d = subtract(a=9, b=7)
print(d) #d = 2

e = subtract(b=5, a=7)
print(e) #e = 2

f = subtract(7, b=5)
print(f) #f = 2

f = subtract(a=7)
print(f) #f = 6

g = subtract(7)
print(g) #g = 6

# h = subtract(b=7, 6)
# print(h) #error
```

```
2
2
2
2
6
6
```

[49]:
```
#Defining functions that can take in ANY number of parameters
def total(*tutorial_marks):
    sum = 0
    for mark in tutorial_marks:
        sum += mark
    return sum

ttl = total(2,2,2,2,2,2,2,2,2,2)
print(ttl) #print 20
```

```
20
```

[50]:
```
car = {
  "brand": "Ford",
  "model": "Mustang",
```

```
  "year": 1964
}

x = car.items()

print(x)
```

```
dict_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 1964)])
```

[51]:
```
def total_mark(**tutorials):
    sum = 0
    for tute, mark in tutorials.items():
        sum += mark
    return sum

my_mark = total_mark(week2=2, week3=2, week4=1, week5=2)
print(my_mark) #print 7
```

```
7
```

[54]:
```
#whatever comment appears after def can help you understand the function's␣
 ↪objective
def find_min(x, y):
    '''This function finds the minimum of two input values'''
    if x <= y:
        return x
    else:
        return y

#help(find_min)
print(find_min.__doc__)
m = find_min(3, 5)
print('Minimum value is ' + str(m))
```

```
This function finds the minimum of two input values
Minimum value is 3
```