CS315

Programming Languages

Fall 2019 Homework 2

Muhammad Arham Khan

21701848

Section 3

12 Dec, 2019

BRIEFING

1. **Test1():** the function is added to show the positional/ keyword and combinational correspondence of parameters. It basically is a normal function that outputs the parameters passed to it in the order they're passed or according to the keywords or both. In some languages where it is not possible to do keyword and positional correspondence using the same function syntax, multiple functions of the name type (1_X) are made that enable keyword and combined parameter correspondence.

2. **Test2():** the function tests the assignment of default values to parameters (automatically assigned values in case that variable isn't provided in the function call).

3. **Test3():** this function tests the possibility of a variable number of parameters in the function call so that the programmer can pass any number of variable to the same function.

4. **Test4():** the function tests the usage of pass-by-value parameter assignment so any parameter is a value copied variable instead of a reference that may be altered in the function.
5. **Test5():** function tests the usage of pass-by-reference parameter assignment so the parameter is a reference to the variable in the dynamic parent and can be altered in the function.

- ## Python:

  - ### CODE:

```
#tests positional correspondence
def test1( a, b, c):
  print( "test1: a=" + a + ", b=" + b + ", c=" + c)
#tests default values
def test2( a = 'defaultA', b = 'defaultB', c = 'defaultC'):
  print( "test2: a=" + a + ", b=" + b + ", c=" + c)
#tests variable number of parameters
def test3( *args):
  print("test3: ", args)
#tests pass by value
def test4( a):
  a = "test4A"
  print( "test4: a=" + a)
#tests pass by reference
def test5( a):
  a.val = "test5A"
  print( "test4: a=", a.val)
```

```
def main():
  #showing positional correspondence
  test1( "a", "b", "c");

  print()
  #showing keyword correspondence
  test1( b = "b", a = "a", c = "c")
  test1( c = "c", b = "b", a = "a")
  test1( a = "a", b = "b", c = "c")

  print()
  #combination of positional and keyword combination
  #_____________________________________________
  #  non-keyword argument after a keyword argument
  #test1( b = "b", a = "a", "c")
  #  duplicate value for the same argument
  #test1( b = "b", "b", "c")

  test1( "a", b = "b", c = "c")
  test1( "a", c = "c", b = "b")

  print()

  #Formal Parameter Default Values
  test2( "a", "b", "c")
  test2( "a", "b")
  test2( "a", c = "c")
  test2()

  print()
  #Variable Number of Actual Parameters
  #____________________________________
  #less than actual number of parameters
  #test1("a", "b") #gives error

  #more than actual parameters
  #test1("a", "b", "c", "d") #gives error

  test3("a", "b");
  test3("a", "b", "c", "d")

  print()
  a = "mainA"
  b = TestParam("b")

  #parameters are passed by value by default in python
  test4( a)
  test5( b)
  #to pass by reference, a wrapper class must be made
  #as only objects are passed by reference
  #so main parameters not changed
  print("main: a=" + a + ", b=" + b.val)

class TestParam:
  def __init__(self, val):
    self.val = val
main()
```

○ **OUTPUT:**

```
test1: a=a, b=b, c=c

test1: a=a, b=b, c=c
test1: a=a, b=b, c=c
test1: a=a, b=b, c=c

test1: a=a, b=b, c=c
test1: a=a, b=b, c=c

test2: a=a, b=b, c=c
test2: a=a, b=b, c=defaultC
test2: a=a, b=defaultB, c=c
test2: a=defaultA, b=defaultB, c=defaultC

test3:  ('a', 'b')
test3:  ('a', 'b', 'c', 'd')
```

```
test4: a=test4A

test4: a= test5A

main: a=mainA, b=test5A
```

- **ANALYSIS:**

Parameter Correspondence (positional, keyword, combination): Python is allows for all three (keyword, positional and combined) parameters correspondence without the need for any extra declarations in the function. Hence, all three types of correspondence are show as working on test() function calls[1].

Formal Parameter Default Values: Python allows for default parameter values and as shown in test2() function calls, it replaces any missing parameters with the default values provided in the function declaration.

Variable Number of Actual Parameters: Python does not allow for a different number of variables to be passed than what have been expected in the function signature and gives errors if there is lower or higher number of parameters passed[1]. But, python does allow the use of arguments list where a variable number of parameters can be passed to and accessed inside a function. So, it is possible to pass a variable number of parameter but that needs a separate type of function (like test3()) signature that can accept variable arguments.

Parameter passing methods: Python, like many other languages, passes primitive parameters by value and there is no way to pass primitive types [1] by reference except if you make a wrapper class for it. Since python passes Objects by reference, use of a wrapper class in test5() function passes the value by reference.

• **<u>Dart:</u>**

- **CODE:**

```
void test1( String a, String b, String c){

  print( "test1: a=" + a + ", b=" + b + ", c=" + c);

}


//enables keyword, combinational correspondence

void test1_1( {String a, String b, String c}){

  print( "test1: a=" + a + ", b=" + b + ", c=" + c);

}


//allows combinational correspondence (a can be positional and b and c can be keywords)

void test1_2( String a, {String b, String c}){

  print( "test1: a=" + a + ", b=" + b + ", c=" + c);

}
```

```
//gives default values to all parameters
void test2( {String a : "defaultA", String b : "defaultB", String c : "defaultC"}){
  print( "test2: a=" + a + ", b=" + b + ", c=" + c);
}

//No test3 function available as variable number of parameters not allowed

//test4 tests pass by value
void test4( String a){
    a = "test4A";
    print( "test4: a=" + a);
}

//test5 tests pass by reference
void test5( TestParam a){
    a.val = "test5A";
    print( "test4: a="+ a.val);
}




void main(){
  //showing positional correspondence
  test1( "a", "b", "c");

  print("\n");
  //showing keyword correspondence
  test1_1( b : "b", a : "a", c : "c");
  test1_1( c : "c", b : "b", a : "a");
  test1_1( a : "a", b : "b", c : "c");

  print("\n");
  //combination of positional and keyword combination
  //_____________________________________________
  // error: named argument expected
  //test1_1( b : "b", a : "a", "c")
  //  error: named argument expected
  //test1_1( "a", b : "b", c: "c")

  test1_2( "a", b : "b", c : "c");
  test1_2( "a", c : "c", b : "b");

  print("\n");

  //Formal Parameter Default Values
  test2( a: "a", b : "b", c :"c");
  test2( a : "a", b : "b");
  test2( a: "a", c : "c");
  test2();

  print("\n");
  //Variable Number of Actual Parameters
  //____________________________________
  //less than or greater than actual number of parameters
  //gives error: incorrect number of arguments passed
  //variable number of arguments not possible
  //test1("a", "b");
  //test1("a", "b", "c", "d");



  print("\n");
  String a = "mainA";
  TestParam b = new TestParam("b");

  //parameters are passed by value by default
  test4( a);
  test5( b);
  //to pass by reference, a wrapper class must be made
  //as only objects are passed by reference
  //so main parameters not changed
  print("main: a=" + a + ", b=" + b.val);
}

class TestParam {
  String val;
  TestParam(this.val);
}
```

○ OUTPUT:

```
test1: a=a, b=b, c=c


test1: a=a, b=b, c=c

test1: a=a, b=b, c=c

test1: a=a, b=b, c=c


test1: a=a, b=b, c=c

test1: a=a, b=b, c=c


test2: a=a, b=b, c=c

test2: a=a, b=b, c=defaultC

test2: a=a, b=defaultB, c=c

test2: a=defaultA, b=defaultB, c=default


test4: a=test4A

test4: a=test5A

main: a=mainA, b=test5A
```

○ ANALYSIS:

Parameter Correspondence (positional, keyword, combination): Dart allows for all three (keyword, positional and combined) parameters correspondence but requires the usage to be declared accordingly in the function signature (during function declarations). [2] So, for keyword correspondence, the parameters must be added to a {} as shown in test1_1 and for combined correspondence, the keyword replaceable variables must be put in {} as shown in test1_2. Hence, all three types of correspondence are possible, but additional code is needed and there is less usage flexbility.

Formal Parameter Default Values: Dart allows for default parameter values and as shown in test2() function calls, it replaces any missing parameters with the default values [2] provided in the function declaration but then function requires keyword correspondence to be used in the function call.

Variable Number of Actual Parameters: Dart does not allow for a different number of variables to be passed than what have been expected in the function signature and gives errors if there is lower or higher number of parameters passed. Similarly, Dart has no way to pass a variable number of arguments (parameters) to the function so test() has not been implemented for this language.

Parameter passing methods: Dart, like many other languages, passes primitive parameters by value and there is no way to pass primitive types by reference [2] except if you make a wrapper class for it. Since dart passes Objects by reference, use of a wrapper class in test5() function passes the value by reference.

- **Javascript:**
  - **CODE:**

```
<!DOCTYPE html>


<head>
        <script type="text/javascript">
                //tests positional correspondence
                function test1(a, b, c) {
                        document.write("test1: a=" + a + ", b=" + b + ", c=" + c +
"<br>");
                }
                //tests default values assignment
                function test2(a = "defaultA", b = "defaultB", c = "defaultC") {
                        document.write("test2: a=" + a + ", b=" + b + ", c=" + c +
"<br>");
                }
                //tests variable number of arguments alottment
                function test3(...args) {
                        document.write("test3: " + args + "<br>");
                }
                //tests pass by value
                function test4(a) {
                        a = "test4A"
                        document.write("test4: a=" + a + "<br>");
                }
                //tests pass by reference
                function test5(a) {
                        a.val = "test5A"
                        document.write("test5: a=" + a.val + "<br>");
                }
                function main() {
                        //showing positional correspondence
                        test1("a", "b", "c");
```

```
document.write("<br>");
//showing keyword correspondence
//keyword correspondence does not work
//ignores labels, takes the parameters
//according to the positions
test1(b = "b", a = "a", c = "c");
test1(c = "c", b = "b", a = "a");
test1(a = "a", b = "b", c = "c");

document.write("<br>");
//combination of positional and keyword combination
//_____________________________________________
// since keyword correspondence not possible,
// combinational follows positional correspondence
test1(b = "b", a = "a", "c");
test1("a", b = "b", c = "c");

document.write("<br>");

//Formal Parameter Default Values
test2("a", "b", "c");
test2("a", "b");
test2("a");
test2();

document.write("<br>");
//Variable Number of Actual Parameters
//____________________________________
//less than or greater than actual number of parameters
//considers missing parameter undefined
test1("a", "b");
//ignores extra parameter
test1("a", "b", "c", "d");
//variable number of parameters accepted
test3("a", "b");
test3("a", "b", "c", "d", "e");

document.write("<br>");
```

```
                        var a = "mainA";

                        var b = { val: 'mainB' };


                        //parameters are passed by value by default

                        test4(a);

                        test5(b);

                        //to pass by reference, an Object must be made

                        //as only objects are passed by reference

                        document.write("main: a=" + a + ", b=" + b.val + "<br>");


                }

                main();

        </script>

</head>


</html>
```

○ OUTPUT:

```
test1: a=a, b=b, c=c

test1: a=b, b=a, c=c
test1: a=c, b=b, c=a
test1: a=a, b=b, c=c

test1: a=b, b=a, c=c
test1: a=a, b=b, c=c

test2: a=a, b=b, c=c
test2: a=a, b=b, c=defaultC
test2: a=a, b=defaultB, c=defaultC
test2: a=defaultA, b=defaultB, c=defaultC

test1: a=a, b=b, c=undefined
test1: a=a, b=b, c=c
test3: a,b
test3: a,b,c,d,e

test4: a=test4A
test5: a=test5A
main: a=mainA, b=test5A
```

○ ANALYSIS:

Parameter Correspondence (positional, keyword, combination): JavaScript allows for only positional parameters correspondence and ignores any attempts for a combined or keyword correspondence, using the positional order of the parameters instead. [3]

Formal Parameter Default Values: JavaScript allows for default parameter values and as shown in test2() function calls, it replaces any missing parameters with the default values provided in the function declaration. But since it does not support keyword correspondence, the language requires consecutive parameters to be provided and cannot default assign any variables in the middle.

Variable Number of Actual Parameters: Javascript allows for a variable number of variables to be passed to a function call and if the number is different than the expected number of variables, Javascript either considers the variable 'undefined' or ignores the extra parameters.[3] javascript does allow the use of arguments list where a variable number of parameters can be passed to and accessed inside a function. So, it is possible to pass a variable number of parameter but that needs a separate type of function (like test3()) signature that can accept variable arguments.

Parameter passing methods: Javascript, like many other languages, passes primitive parameters by value and there is no way to pass primitive types by reference except if you make a wrapper object for it. Since javascript passes Objects by reference, use of an object in test5() function passes the value by reference.

- **PHP:**
  - **CODE:**

```
<!DOCTYPE html>
  <head>
    <title>Php Parameters</title>
  </head>
  <body>
    <?php
      //tests positional correspondence
      function test1( $a, $b, $c){
        echo "test1: a=$a, b=$b, c=$c\n";
      }
      //tests default values
      function test2( $a = "defaultA", $b = "defaultB", $c = "defaultC"){
        echo "test1: a=$a, b=$b, c=$c\n";
      }
      //tests variable number of parameters
      function test3(){

          $args = func_get_args();
          echo "test3: ";
          foreach ($args as $arg){
            echo $arg.", ";
          }
          echo "\n";
      }
      //tests pass by value
      function test4( $a ){
          $a = "test4A";
        echo "test4: a=$a\n";
      }
      //tests pass by reference
      function test5( &$a ){
          $a = "test5A";
        echo "test5: a=$a\n";
      }
      function main(){
        //showing positional correspondence
        test1( "a", "b", "c");

        echo("\n");
        //showing keyword correspondence
          //keyword correspondence does not work
          //ignores labels, takes the parameters
          //according to the positions
        test1( $b = "b", $a = "a", $c = "c");
        test1( $c = "c", $b = "b", $a = "a");
        test1( $a = "a", $b = "b", $c = "c");

        echo "\n";
          //combination of positional and keyword combination
          //_____________________________________________
          // since keyword correspondence not possible,
          // combinational follows positional correspondence
          test1($b = "b", $a = "a", "c");
          test1("a", $b = "b", $c = "c");

          echo "\n";

          //Formal Parameter Default Values
          test2("a", "b", "c");
          test2("a", "b");
```

```
            test2("a");
            test2();

            echo "\n";
            //Variable Number of Actual Parameters
            //____________________________________
            //less than or greater than actual number of parameters
            //error: too few arguments error
            //test1("a", "b");
            //ignores extra parameter
            test1("a", "b", "c", "d");

            //variable number of parameters accepted
            test3("a", "b");
            test3("a", "b", "c", "d", "e");

            echo "\n";
            $a = "mainA";
            $b = "mainB";

            //parameters are passed by value by default
            test4($a);
            test5($b);
            //to pass by reference, add & vefore variable
            echo "main: a=" . $a . ", b=" . $b . "\n";

        }

        main();
    ?>
    </body>
  </html>
```

- **OUTPUT:**

```
test1: a=a, b=b, c=c

test1: a=b, b=a, c=c
test1: a=c, b=b, c=a
test1: a=a, b=b, c=c
```

```
test1: a=b, b=a, c=c
test1: a=a, b=b, c=c

test1: a=a, b=b, c=c
test1: a=a, b=b, c=defaultC
test1: a=a, b=defaultB, c=defaultC
test1: a=defaultA, b=defaultB, c=defaultC

test1: a=a, b=b, c=c
test3: a, b,
test3: a, b, c, d, e,

test4: a=test4A
test5: a=test5A
main: a=mainA, b=test5A
```

- **ANALYSIS:**

Parameter Correspondence (positional, keyword, combination): PHP allows for only positional parameters correspondence and ignores any attempts for a combined or keyword correspondence, using the positional order of the parameters instead.

Formal Parameter Default Values: PHP allows for default parameter values and as shown in test2() function calls, it replaces any missing parameters with the default values provided in the function declaration. But since it does not support keyword correspondence[4], the language requires consecutive parameters to be provided and cannot default assign any variables in the middle.

Variable Number of Actual Parameters: PHP partially allows for a variable number of variables to be passed to a function call and if the number is lower than the expected number of variables, PHP gives a lower than expected number of arguments error but if the passed number of parameters is higher than the expected value, it ignores the extra variable and does not throw an exception. PHP does allow the use of arguments list where a variable number of parameters can be passed to and accessed inside a function. So, it is possible to pass a variable number of parameter but that needs a separate type of function (like test3()) that can accept variable arguments.

Parameter passing methods: Php allows the passing of variables by value and by reference based upon the need. So, by default the variables are passed by value but if a variable needs to be passed by reference, a ‘&’ can be added in the function signature to make it pass by reference.[4]

- **<u>Perl:</u>**
  - **CODE:**

```
sub test1{
  my ($a, $b, $c) = @_;
  print "test1: a=" . $a . ", b=" . $b . ", c=" . $c . "\n";
}

#test1_1 allows for keyword correspondence
sub test1_1{
  %vals = @_;
  my ($a, $b, $c) = ($vals{a}, $vals{b}, $vals{c});
  print "test1: a=" . $a . ", b=" . $b . ", c=" . $c . "\n";
}
#allows combinational correspondence
sub test1_2{
  ($a, %vals) = @_;
  my ( $b, $c) = ( $vals{b}, $vals{c});
  print "test1: a=" . $a . ", b=" . $b . ", c=" . $c . "\n";
}

#adds default values to parameters
sub test2{
  %vals = (
    'a' => 'defaultA',
    'b' => 'defaultB',
    'c' => 'defaultC',
    @_
  );
  my ($a, $b, $c) = ($vals{a}, $vals{b}, $vals{c});
  print "test2: a=" . $a . ", b=" . $b . ", c=" . $c . "\n";
}

#can input variable number of parameters
sub test3{
  print "test3: ";
  for( @_){
    print  $_ . ",";
  }
  print "\n";
}
```

```
#simulates pass-by-value
sub test4{
  #assigning variable to another variable to copy
  my ($a) = @_;
  $a = "test4A";
  print "test4: a=" . $a . "\n";
}

#edits variable in static parent because passed
#by reference
sub test5{
  @_[0] = "test5A";
  print "test5: a=" . @_[0] . "\n";
}

sub main(){
  #showing positional correspondence
  test1( "a", "b", "c");

  print "\n";
  #showing keyword correspondence
  test1_1( b => "b", a => "a", c => "c");
  test1_1( c => "c", b => "b", a => "a");
  test1_1( a => "a", b => "b", c => "c");

  print "\n";
  #combination of positional and keyword combination
  #_____________________________________________
  #  not allowed: leaves elements without keyword assignment as null
  test1_1( b => "b", a => "a", "c");
  test1_1( b => "b", "b", "c");

  #allows combinational correspondence
  # a is passed by position, b and c are passed by keywords
  test1_2( "a", b => "b", c => "c");
  test1_2( "a", c => "c", b => "b");

  print "\n";
  #Formal Parameter Default Values
  #Default assignment only works on
  #keyword correspondence
  test2( a => "a", b => "b");
  test2( a => "a");
  test2( b => "b", c => "c");
  test2();



  print "\n";
  #Variable Number of Actual Parameters
  #____________________________________
  #allowed, but unassigned parameters are null
  test1("a", "b");
  #allowed, but extra parameters are ignored
  test1("a", "b", "c", "d");

  test3("a", "b");
  test3("a", "b", "c", "d");



  print "\n";
  $a = "mainA";
  $b = "mainB";

  #parameters are passed by reference by default in python
  test4( $a);
  test5( $b);
  #to pass by value, the passed parameter cannot be used
  #in the subroutine as it is passed by reference
  print "main: a=" . $a . ", b=" . $b . "\n";

}

main();
```

- **OUTPUT:**

```
test1: a=a, b=b, c=c


test1: a=a, b=b, c=c

test1: a=a, b=b, c=c

test1: a=a, b=b, c=c


test1: a=a, b=b, c=

test1: a=, b=c, c=

test1: a=a, b=b, c=c

test1: a=a, b=b, c=c


test2: a=a, b=b, c=defaultC

test2: a=a, b=defaultB, c=defaultC

test2: a=defaultA, b=b, c=c

test2: a=defaultA, b=defaultB, c=defaultC


test1: a=a, b=b, c=

test1: a=a, b=b, c=c

test3: a,b,

test3: a,b,c,d,


test4: a=test4A

test5: a=test5A

main: a=mainA, b=test5A
```

ANALYSIS:

Parameter Correspondence (positional, keyword, combination): Perl allows for all three (keyword, positional and combined) parameters correspondence but requires the usage to be declared accordingly in the function signature (during function declarations). So, for keyword correspondence, the parameters must be passed to a hashmap (%XXX) from where it can be obtained using its name as shown in test1_1 and for combined correspondence, the keyword replaceable variables must be put in a hashmap(%XXX) and the positional correspondence variable should be passed and received as it like shown in test1_2.[5] Hence, all three types of correspondence are possible, but additional code is needed and there is less usage flexibility.

Formal Parameter Default Values: Perl allows for default parameter values and as shown in test2() function calls, it replaces any missing parameters with the default values provided in the function declaration but then function requires keyword correspondence to be used in the function call as shown in test2() usage.[5]

Variable Number of Actual Parameters: Since perl passes all parameters in a list of arguments, Perl allows for a different number of variables to be passed than what have been expected in the function signature and does not give errors and if there is lower or higher number of parameters passed, it either makes the missing variable null or ignores the extra parameters. Similarly, Perl also allows to pass a variable number of arguments (parameters) to the function so test3() shows that a variable number of parameters can be passed and sued in a function.

Parameter passing methods: since Perl passes parameters in a list by references of the actual parameters, if the @_ (argument list) is mapped, it is copied to a local variable and can be used as if it was passed by value but If the argument list index is edited, it will be as if the variable was passed by reference and the variable in dynamic parent will be edited.[5] Hence, both types of passing methods are possible.

References

[1] “Official Python Documentation”

https://docs.python.org/3/ [Accessed Dec 12, 2019]

[2] “Official Dart Documentation”

https://dart.dev/guides [Accessed Dec 12, 2019]

[3] “Mozilla Javascript Documentation”

https://developer.mozilla.org/en-US/docs/Web/JavaScript [Accessed Dec 12, 2019]

[4] “Official PHP Documentation”

https://www.php.net/docs.php [Accessed Dec 12, 2019]

[5] “Official Perl Documentation”

https://perldoc.perl.org/ [Accessed Dec 12, 2019]