ADVANCED LEVEL EXAMINATION

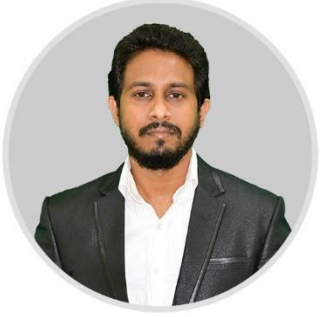
INFORMATION AND COMMUNICATION TECHNOLOGY (ICT)

WEBSITE
WWW.ALICT.EDU.LK

UNIT-09

COMPUTER PROGRAMMING

පරිගණක ක්‍රමලේඛකරණය



AMILA JAYASENA
B.SC. IN INFORMATION TECHNOLOGY
071 831 9024

Table of content

15. Algorithms
    i. Flow charts
    ii. Pseudo codes
    iii. Hand traces

16. Evolution of programming languages
    i. Programming paradigms o Imperative languages
        a) Declarative languages
        b) Object oriented languages

17. Need of program translation
    i. Source program
    ii. Object program
    iii. Program translators
        a) Interpreters
        b) Compilers
        c) Hybrid approach
    iv. Linkers

18. Uses problem-solving process
    i. Understanding the problem
    ii. Defining the problem and boundaries
    iii. Planning solution
    iv. Implementation
- Modularization
- Top-down design and stepwise refinement
- Structure charts

19. Basic features of IDE
    i. Instructions to use
        a) Opening and saving files
        b) Compiling,
        c) executing programs
    ii. Debugging facilities

Section 1

1. First Python Program

Python code	Output
print ("Hello, World!")	

2. Python Comments

Single Line Comments

```
#This is a comment
print ("Hello, World!")
```

```
print ("Hello, World!") #This is a comment
```

```
#print ("Hello, World!")
```

Multi Line Comments

```
#This is a comment
#written in
#More than just one line
print ("Hello, World!")
```

```
"""
This is a comment
written in
more than just one line
"""
print ("Hello, World!")
```

3. Python Variables

- Variables

Python code	output
x = 5 y = "John" print(x) print(y)	

Python code	output
x = 4 x = "Sally" print(x)	

- Casting

Python code	output
x = str (3) y = int (3) z = float (3) print(x) print(y) print(z)	

- Get the Datatype

Python code	output
x = 5 y = "John" print(type(x)) print(type(y))	

- Single or Double Quotes?

Python code	output
x = "John" y = 'John' print(x) print(y)	

- Case-Sensitive

Python code	output
a = 4 A = "Sally" print(a) print(A)	

Python - Variable Names

- විචල්‍ය නාමයක් අකුරකින් හෝ යටි ඉරි අකුරින් ආරම්භ විය යුතුය
- විචල්‍ය නාමයක් අංකයකින් ආරම්භ කළ නොහැක
- විචල්‍ය නාමයක අඩංගු විය හැක්කේ ඇල්ෆා-සංඛ්‍යා අක්ෂර සහ යටි ඉරි (A-z, 0-9, සහ _) පමණි
- විචල්‍ය නම් අවස්ථා-සංවේදී වේ (වයස, වයස සහ AGE යනු විවිධ විචල්‍ය තුනකි)

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)

Variable names:	Correct or incorrect
myvar = "John"	
my_var = "John"	
_my_var = "John"	
myVar = "John"	
MYVAR = "John"	
myvar2 = "John"	
2myvar = "John"	
my-var = "John"	
my var = "John"	

- Multi Words Variable Names

Camel Case	Pascal Case	Snake Case
myVariableName = "John"	MyVariableName = "John"	my_variable_name = "John"

- Many Values to Multiple Variables

Python code	output
x, y, z = "Orange", "Banana", "Cherry" print(x) print(y) print(z)	

- One Value to Multiple Variables

Python code	output
x = y = z = "Orange" print(x) print(y) print(z)	

- Python - Output Variables

Python code	output
x = "awesome" print ("Python is " + x)	

Python code	output
x = "Python is " y = "awesome" z = x + y print(z)	

Python code	output
x = 5 y = 10 print (x + y)	

Python code	output
x = 5 y = "John" print (x + y)	

- Python - Global and local Variables

Python code	output
x = "awesome" def myfunc(): print("Python is " + x) myfunc()	

Python code	output

x = "awesome"
def myfunc():
  x = "fantastic"
  print("Python is " + x)
myfunc()
print("Python is " + x)	

4. Python Data Types

Text	**str**
Numeric	int, float
Sequence	list, tuple, range
Mapping	dict
Set	set
Boolean	bool

- Get the Data Type

Python code	output
x = 5 print(type(x))	

- Setting the Data Type

Python code	output
x = "Hello World" print(type(x))	
x = 20 print(type(x))	
x = 20.5 print(type(x))	
x = ["apple", "banana", "cherry"] print(type(x))	
x = ("apple", "banana", "cherry") print(type(x))	

x = range (6) print(type(x))	
x = {"name”: "John", "age”: 36} print(type(x))	
x = {"apple", "banana", "cherry"} print(type(x))	
x = True print(type(x))	

- Setting the Specific Data Type

Python code	output
x = str("Hello World") print(type(x))	
x = int (20) print(type(x))	
x = float (20.5) print(type(x))	

- The global Keyword

Python code	output

def myfunc():
  global x
  x = "fantastic"
 	
myfunc()
 	
print("Python is " + x)	

x = "awesome"
def myfunc():
  global x
  x = "fantastic"
myfunc()
 	
print("Python is " + x)	

Python Numbers

- Type Conversion

Python code	output
x = 1 y = 2.8 a = float(x) b = int(y) print(a) print(b) print(type(a)) print(type(b))	

- Random Number

Python code	output
import random print(random.randrange(1, 50))	

Python Casting

Python code	output
x = int (1) y = int (2.8) z = int ("3") print(x) print(y) print(z)	
x = float (1) y = float (2.8) z = float ("3") w = float ("4.2") print(x) print(y) print(z)	
x = str("s1") y = str (2) z = str (3.0) print(x) print(y) print(z)	

Python Strings

Python code	output
print("Hello") print('Hello')	

Python code	output
a = "Hello" print(a)	

- Multiline Strings

Python code	output
a = """Python is a high-level, general-purpose programming language.""" print(a)	
a = '''Python is a high-level, general-purpose programming language. ''' print(a)	

- Strings are Arrays

Python code	output
a = "Hello, World!" print(a[1])	

- Looping Through a String

Python code	output
for x in "banana": print(x)	

- String Length

Python code	output
a = "Hello, World!" print(len(a))	

- Check String

Python code	output
txt = "The best things in life are free!" print("free" in txt)	
txt = "The best things in life are free!" if "free" in txt: print("Yes, 'free' is present.")	

- Check if NOT

Python code	output
txt = "The best things in life are free!" print("expensive" not in txt)	
txt = "The best things in life are free!" if "expensive" not in txt: print("Yes, 'expensive' is NOT present.")	

- Python - Slicing Strings

Python code	output
b = "Hello, World!" print(b[2:5])	

- Slice from the Start

Python code	output
b = "Hello, World!" print(b[:5])	

- Slice to the End

Python code	output
b = "Hello, World!" print(b[2:])	

- Negative Indexing

Python code	output
b = "Hello, World!" print(b[-5:-2])	

- Upper Case

Python code	output
a = "Hello, World!" print(a.upper())	

- Split

Python code	output
a = "Hello, World!" print(a.split("l"))	

- String Concatenation

Python code	output
a = "Hello" b = "World" c = a + b print(c)	
a = "Hello" b = "World" c = a + " " + b print(c)	

- Python - Format - Strings

Python code	output
age = 36 txt = "My name is John, I am " + age print(txt)	

Python code	output
age = 36 txt = "My name is John, and I am {}" print(txt.format(age))	

Python code
quantity = 3 itemno = 567 price = 49.95 myorder = "I want {} pieces of item {} for {} dollars." print(myorder.format(quantity, itemno, price))
output

Python code
quantity = 3 itemno = 567 price = 49.95 myorder = "I want to pay {2} dollars for {0} pieces of item {1}." print(myorder.format(quantity, itemno, price))
output

- Python - Escape Characters

Python code
txt = "We are the so-called "Vikings" from the north." print(txt)
txt = "We are the so-called \"Vikings\" from the north." print(txt)
txt = "We are the so-called \\Vikings\\ from the north." print(txt)
txt = "We are the so-called \nVikings\n from the north." print(txt)
txt = "We are the so-called \rVikings\r from the north." print(txt)
txt = "We are the so-called \tVikings\t from the north." print(txt)
txt = "We are the so-called \bVikings\b from the north." print(txt)

- Escape Characters

\'	Single Quote
****	Backslash
\n	New Line
\r	Carriage Return
\t	Tab
\b	Backspace

Python Booleans

Python code	output
print (10 > 9) print (10 == 9) print (10 < 9) print (True / True) print (False / False) print (True / False) print (False / True)	

Python code	output

a = 200
b = 33
 	
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")	

Python code	output
print(bool("Hello")) print (bool (15))	

Python code	output
x = "Hello" y = 15 print(bool(x)) print(bool(y))	

Python code	output
bool("abc") bool (123) bool (["apple", "cherry", "banana"])	

Python code	output
bool (False) bool (None) bool (0) bool ("") bool (()) bool ([]) bool ({})	

- Functions can Return a Boolean

Python code	output

class myclass():
  def __len__(self):
  return 0
 	
myobj = myclass()
print(bool(myobj))	

Python code	output
def myFunction() : return True print(myFunction())	

Python code	output

def myFunction() :
  return True
 	
if myFunction():
  print("YES!")
else:
  print("NO!")	

Python code	output
x = 200 print(isinstance(x, int))	

5. Python Operators

1) Arithmetic operators
2) Assignment operators
3) Comparison operators
4) Logical operators
5) Identity operators
6) Membership operators
7) Bitwise operators

1) Python Arithmetic Operators

Operator	Description	Example
+	Addition	2 + 3 (returns 5)
-	Subtraction	5 - 2 (returns 3)
*	Multiplication	2 * 3 (returns 6)
/	Division	10 / 3 (returns 3.3333)
//	Floor Division	10 // 3 (returns 3)
%	Modulo	10 % 3 (returns 1)
**	Exponentiation	2 ** 3 (returns 8)

Python code	output
x = 5 y = 3 print (x + y)	
x = 5 y = 3 print (x - y)	
x = 5 y = 3 print (x * y)	
x = 12 y = 3 print (x / y)	
x = 5 y = 2 print (x % y)	
x = 2 y = 5 print (x ** y	
x = 15 y = 2 print (x // y)	rounds the result down to the nearest whole number

2) Python Assignment Operators

Operator	Description	Example
=	Assigns the value to a variable	x = 5 (x is assigned the value 5)
+=	Adds the value and assigns it to a variable	x += 3 (x is incremented by 3)
-=	Subtracts the value and assigns it to a variable	x -= 2 (x is decremented by 2)
*=	Multiplies the value and assigns it to a variable	x *= 4 (x is multiplied by 4)
/=	Divides the value and assigns it to a variable	x /= 2 (x is divided by 2)
//=	Performs floor division and assigns it to a variable	x //= 3 (x is floor divided by 3)
%=	Calculates the modulus and assigns it to a variable	x %= 2 (x is assigned the remainder of x divided by 2)
**=	Performs exponentiation and assigns it to a variable	x **= 3 (x is raised to the power of 3)
&=	Performs bitwise AND and assigns it to a variable	x &= 3 (x is bitwise ANDed with 3)
`	=`	Performs bitwise OR and assigns it to a variable
^=	Performs bitwise XOR and assigns it to a variable	x ^= 2 (x is bitwise XORed with 2)
>>=	Performs right shift and assigns it to a variable	x >>= 1 (x is right-shifted by 1)
<<=	Performs left shift and assigns it to a variable	x <<= 2 (x is left-shifted by 2)

Python code	output
x = 5 x += 3 print(x)	
x = 5 x -= 3 print(x)	
x = 5 x *= 3 print(x)	
x = 5 x /= 3 print(x)	

x = 5 x%=3 print(x)	
x = 5 x//=3 print(x)	
x = 5 x//=3 print(x)	
x = 5 x &= 3 print(x)	
x = 5 x |= 3 print(x)	
x = 5 x ^= 3 print(x)	
x = 5 x >>= 3 print(x)	
x = 5 x <<= 3 print(x)	

3) Python Comparison Operators

Operator	Description	Example
==	Equal to	x == y (returns True if x is equal to y, otherwise False)
!=	Not equal to	x != y (returns True if x is not equal to y, otherwise False)
>	Greater than	x > y (returns True if x is greater than y, otherwise False)
<	Less than	x < y (returns True if x is less than y, otherwise False)
>=	Greater than or equal to	x >= y (returns True if x is greater than or equal to y, otherwise False)
<=	Less than or equal to	x <= y (returns True if x is less than or equal to y, otherwise False)

Python code	output
x = 5 y = 3 print (x == y)	
x = 5 y = 3 print (x != y)	
x = 5 y = 3 print (x > y)	
x = 5 y = 3 print (x < y)	
x = 5 y = 3 print (x >= y)	
x = 5 y = 3 print (x <= y)	

4) Python Logical Operators

Operator	Description	Example
and	Returns True if both operands are True, otherwise False	x and y
or	Returns True if either operand is True, otherwise False	x or y
not	Returns the opposite boolean value of the operand	not x

Python code	output
x = 5 print (x > 3 and x < 10)	
x = 5 print (x > 3 or x < 4)	
x = 5 print (not(x > 3 and x < 10))	

5) Python Identity Operators

Operator	Description	Example
is	Returns True if the operands are the same object, otherwise False	x is y
is not	Returns True if the operands are not the same object, otherwise False	x is not y

Python code	output
x = ["apple", "banana"] y = ["apple", "banana"] z = x print(x is z) print(x is y) print(x == y)	
x = ["apple", "banana"] y = ["apple", "banana"] z = x print(x is not z) print(x is not y) print(x != y)	

6) Python Membership Operators

Operator	Description	Example
in	Returns True if the value or element is found in the sequence or collection, otherwise False	x in y
not in	Returns True if the value or element is not found in the sequence or collection, otherwise False	x not in y

Python code	output
x = ["apple", "banana"] print("banana" in x)	
x = ["apple", "banana"] print("pineapple" not in x)	

7) Python Bitwise Operators

Operator	Description	Example
&	Bitwise AND	x & y
|	Bitwise OR	x | y
^	Bitwise XOR	x ^ y
~	Bitwise NOT	~x
<<	Left shift	x << n
>>	Right shift	x >> n

Python code	output
a = 60 b = 13 c = 0 c = a & b print (c)	
c = a | b print (c)	
c = a ^ b print (c)	
c = ~a print (c)	
c = a << 2 print (c)	
c = a >> 2 print (c)	

6. Python operator precedence

I. Here is the general precedence and associativity of operators in Python, from highest to lowest:

1. Parentheses: ( )
2. Exponentiation: **
3. Unary operators: +x, -x, ~x
4. Multiplication, Division, and Remainder: *, /, //, %
5. Addition and Subtraction: +, -
6. Bitwise Shifts: <<, >>
7. Bitwise AND: &
8. Bitwise XOR: ^
9. Bitwise OR: |
10. Comparison Operators: ==, !=, >, <, >=, <=, is, is not, in, not in
11. Logical NOT: not
12. Logical AND: and
13. Logical OR: or

```
result1 = 2 + 3 * 4       # Result: 14 (Multiplication has higher precedence than addition)
result2 = (2 + 3) * 4     # Result: 20 (Parentheses override precedence)
result3 = 2 + 3 ** 2      # Result: 11 (Exponentiation has higher precedence than addition)
result4 = (2 + 3) ** 2    # Result: 25 (Parentheses override precedence)
result5 = 2 * 3 + 4 / 2   # Result: 8 (Multiplication and division have equal precedence, evaluated left-to-right)
result6 = 2 * (3 + 4) / 2 # Result: 7 (Parentheses override precedence)

print(result1)
print(result2)
print(result3)
print(result4)
print(result5)
print(result6)
```

Section 2