



NÚMEROS

PROGRAMACIÓN DE COMPUTADORAS III

Abdel G. Martínez L.

CATEGORÍAS

⦿ *Enteros*

- Números positivos o negativos sin cifras decimales
- Pueden ser de un tipo: `int`

```
>>> ent = 35
```

- Antes existían los `long`

CATEGORÍAS

⦿ *Enteros*

- Otro uso es para expresar números en octal o hexadecimal

```
>>> oct = 0o20
```

```
>>> hex = 0x17
```

CATEGORÍAS

⦿ *Reales*

- Números que tienen decimales
- Puede ser de un tipo: `float`
- Se expresan de dos maneras:
 - De forma tradicional, con una parte entera y una decimal

```
>>> pi = 3.1416
```

- Utilizando un exponente de base 10

```
>>> celula = 0.1e-5
```

CATEGORÍAS

⦿ *Complejos*

- Números que tienen una parte imaginaria
- Si no los conoce, quizás no tenga que utilizarlos
- Los puede reconocer porque tienen con una j al final
- Su uso más común es en la geología, ingeniería y física

```
>>> latitud = 15 + 3j
```



OPERADORES ARITMÉTICOS

PROGRAMACIÓN DE COMPUTADORAS III

Abdel G. Martínez L.

CONCEPTO Y CATEGORÍAS

- ⦿ Operaciones que toman y retornan valores numéricos

Operador	Nombre	Descripción	Ejemplo
+	Suma	Suma dos valores	$3 + 4$
-	Resta	Resta dos valores	$5 - 2$
*	Multiplicación	Multiplica dos valores	$8 * 7$
**	Exponenciación	Eleva un valor a otro valor	$2 ** 3$
/	División	Divide dos valores, retorna real	$16 / 2$
//	División Entera	Divide dos valores, retorna entero	$15 // 2$
%	División Modular	Divide dos valores, retorna residuo	$5 \% 3$



PROGRAMACIÓN DE COMPUTADORAS III

Abdel G. Martínez L.

CONCEPTO Y DEFINICIÓN

- ⦿ Textos encerrados entre apóstrofes (') o comillas (")

```
>>> nombre = 'Carlos'
```

```
>>> color = "azul"
```

- ⦿ Para texto en múltiples líneas, se utiliza la triple comilla (""")

```
>>> frase = """El ignorante afirma,
```

```
... el sabio duda y reflexiona."""
```

OPERACIONES

- ⦿ Para concatenar usamos el operador de suma

```
>>> a = "Hola "
```

```
>>> b = "Mundo"
```

```
>>> print(a + b)
```

- ⦿ Para repetir una cadena, usamos el operador de multiplicación

```
>>> print(a * 3)
```



BOOLEANOS

PROGRAMACIÓN DE COMPUTADORAS III

Abdel G. Martínez L.

CONCEPTO Y DEFINICIÓN

- ⦿ Representan valores de lógica binaria
- ⦿ Pueden ser verdaderos o falsos
 - >>> `condicion = True`
 - >>> `condicion = False`
- ⦿ Son útiles para ciclos y condiciones



OPERADORES LÓGICOS

PROGRAMACIÓN DE COMPUTADORAS III

Abdel G. Martínez L.

CONCEPTO Y CATEGORÍAS

- ⦿ Evalúa que se cumpla o no una condición
- ⦿ Proporciona un resultado booleano

Operador	Descripción	Ejemplo
AND	Ambas expresiones deben ser ciertas para obtener un resultado cierto	>>> True and False False
OR	Una expresión debe ser cierta para obtener un resultado cierto	>>> True or False True
NOT	Invierte el valor booleano (si es True, lo vuelve False y viceversa)	>>> not True False



OPERADORES RELACIONALES

PROGRAMACIÓN DE COMPUTADORAS III

Abdel G. Martínez L.

CONCEPTO Y CATEGORÍAS

- ◉ Símbolos para comparar dos valores
- ◉ El resultado es un tipo de dato booleano
- ◉ Si el resultado de la comparación es correcta, entonces la expresión es verdadera. De lo contrario, la expresión es falsa.

CONCEPTO Y CATEGORÍAS

Operador	Nombre	Descripción	Ejemplo
==	Igual	Verifica si ambos valores son idénticos	>>> 17 == 20 False
!=	Distinto	Verifica si ambos valores son distintos	>>> 17 != 20 True
<	Menor que	Verifica que el primer valor sea menor que el segundo valor	>>> 17 < 20 True
>	Mayor que	Verifica que el primer valor sea mayor que el segundo valor	>>> 17 > 20 False
<=	Menor o igual	Verifica que el primer valor sea igual o menor que el segundo valor	>>> 17 <= 20 True
>=	Mayor o igual	Verifica que el primer valor sea igual o mayor que el segundo valor	>>> 17 >= 20 False

```

PPOptional.add_argument('-v', dest='verbose', action='count', default=0, help='increase verbosity level')
PPOptional.add_argument('--version', version=Version(' ' + str(constant.CURRENT_VERSION)), help='laZagne version')

PWrite = argparse.ArgumentParser(add_help=False, formatter_class=lambda prog: argparse.HelpFormatter(prog, max_help_position=constant.MAX_HELP_POSITION))
PWrite._optionals.title = 'output'
PWrite.add_argument('-w', dest='write', action='store_true', help='write a text file on the current directory')

all_subparser = []
for c in category.keys():
    category[c]['parser'] = argparse.ArgumentParser(add_help=False, formatter_class=lambda prog: argparse.HelpFormatter(prog, max_help_position=constant.MAX_HELP_POSITION))
    category[c]['parser']._optionals.title = category[c]['help']

    category[c]['subparser'] = []
    for module in modules[c].keys():
        m = modules[c][module]
        category[c]['parser'].add_argument(m.options['command'], dest=m.options['dest'], action=m.options['action'], help=m.options['help'])

        if m.suboptions and m.name != 'thunderbolt':
            tmp = []
            for sub in m.suboptions:
                tmp_subparser = argparse.ArgumentParser(add_help=False, formatter_class=lambda prog: argparse.HelpFormatter(prog, max_help_position=constant.MAX_HELP_POSITION))
                tmp_subparser._optionals.title = sub['title']
                if 'type' in sub:
                    tmp_subparser.add_argument(sub['command'], dest=sub['dest'], action=sub['action'], help=sub['help'])
                else:
                    tmp_subparser.add_argument(sub['command'], dest=sub['dest'], help=sub['help'])
                tmp.append(tmp_subparser)
            all_subparser.append(tmp_subparser)
        category[c]['subparser'] += tmp

parents = [PPOptional] + all_subparser + [PWrite]
dic = {'all': {'parents': parents, 'help': 'Run all modules', 'func': 'run_all_modules'}}
for c in category.keys():
    parser_tab = [PPOptional, category[c]['parser']]
    if 'subparser' in category[c]:
        if category[c]['subparser']:
            parser_tab += category[c]['subparser']
    parser_tab += [PWrite]
    dic_tmp = {c: {'parents': parser_tab, 'help': 'Run %s module' % c, 'func': 'run_module' % c}}
    dic = dict(dic.items() + dic_tmp.items())

subparsers = parser.add_subparsers(help='Choose a main command')
for d in dic.keys():
    subparsers.add_parser(d, parents=dic[d]['parents'], help=dic[d]['help']).set_defaults(func=dic[d]['func'], auditType=d)

```

VARIABLES

PROGRAMACIÓN DE COMPUTADORAS III

Abdel G. Martínez L.

CONCEPTO Y DEFINICIÓN

- ⦿ Espacios en memoria
- ⦿ Almacenan un valor cambiante
- ⦿ Se identifican con un nombre simbólico

REGLAS

- ⦿ Iniciar con una letra (a-z, A-Z) o con el carácter subrayado (_)
- ⦿ Los caracteres siguientes son letras, números o subrayado (_)
- ⦿ Sensitivo a las mayúsculas y minúsculas
- ⦿ Longitud razonable
- ⦿ Cuidado con las palabras reservadas

CONSIDERACIONES ESPECIALES

- ⦿ Los tipos de datos de las variables no son fijos

```
>>> a = "Analida"
```

```
>>> a = True
```

```
>>> a = 33
```



ENTRADA DE DATOS

PROGRAMACIÓN DE COMPUTADORAS III

Abdel G. Martínez L.

CONCEPTO Y DEFINICIÓN

- ⦿ Existen múltiples formas de generar datos en un programa
- ⦿ Base de datos, fichero, clics del ratón o de Internet
- ⦿ La forma más común es a través del teclado

CONCEPTO Y DEFINICIÓN

- ⦿ La función `input()` es la solución de Python para esta situación
 - Detiene el flujo de ejecución hasta que el usuario escriba un texto y se presione la tecla ENTER
 - Admite un parámetro, el cual se imprime en pantalla como ayuda para el usuario final

CONSIDERACIONES ESPECIALES

- ⦿ Todo texto de entrada es una cadena
- ⦿ Si se desea transformar a otro tipo de dato, debe utilizarse otros algoritmo y funciones



SALIDA DE DATOS

PROGRAMACIÓN DE COMPUTADORAS III

Abdel G. Martínez L.

CONCEPTO Y DEFINICIÓN

- ⦿ Cada programa debe poder comunicarse con su entorno
- ⦿ Para este propósito nacieron las funciones de entrada/salida
- ⦿ En el video anterior aprendieron cómo ingresar datos
- ⦿ Imprimiremos los resultados de nuestros programas en pantalla

CONCEPTO Y DEFINICIÓN

- ⦿ La función `print()` es la solución de Python para esta situación

- Permite imprimir directamente un texto

```
>>> print("Hola Eduvolucion!")
```

```
Hola Eduvolucion!
```

- Puede imprimir el valor de una variable

```
>>> x = 3
```

```
>>> print(x)
```

```
3
```

CONCEPTO Y DEFINICIÓN

⦿ La función `print()` es la solución de Python para esta situación

- Podemos imprimir varios valores separados por coma

```
>>> a = 5
```

```
>>> b = 7
```

```
>>> print(a, b)
```

- Permite mezclar impresión de textos y valores de variables

```
>>> nombre = "Carlos"
```

```
>>> print("Tu nombre es " + nombre)
```


CONSIDERACIONES ESPECIALES

- ⦿ Si la variable es de tipo numérica, debemos utilizar la función `str()` para convertir su valor en una cadena

```
>>> edad = 35
```

```
>>> print("Tu edad es " + str(edad))
```



COMENTARIOS

PROGRAMACIÓN DE COMPUTADORAS III

Abdel G. Martínez L.

CONCEPTO Y DEFINICIÓN

- ◉ Un comentario es una anotación incrustada en el código fuente
- ◉ Legible para el programador
- ◉ Ignorada por el intérprete

PROPÓSITO

- ⦿ Hacer el código fuente más fácil de entender
- ⦿ Útil para el mantenimiento y reutilización de código

OTROS USOS

- ⦿ Generación de documentación externa
- ⦿ Integración con un sistema de control de versiones
- ⦿ Para el proceso de depuración

CATEGORÍAS

- ◉ *Una línea*

- Utilizamos el numeral al inicio de la línea a documentar

```
>>> print("No es comentario")
```

```
>>> #print("Comentario")
```

CATEGORÍAS

⦿ *Múltiples líneas*

- Usamos tres apóstrofes al inicio y final de la parte a documentar

```
>>> '''
```

```
... print("Comentario 1")
```

```
... print("Comentario 2")
```

```
... '''
```

```
>>> print("Sin comentar")
```




SECUENCIAS DE ESCAPE

PROGRAMACIÓN DE COMPUTADORAS III

Abdel G. Martínez L.

CONCEPTO Y DEFINICIÓN

- ◉ Conjunto de caracteres en textos que son interpretados con alguna finalidad en particular
- ◉ Son útiles cuando tratamos de imprimir un texto y no se pueden incluir ciertos caracteres
- ◉ Inician con barra inversa y son interpretadas diferentemente

```
>>> print("Y Antonio dijo: \"¿Dónde fue?\"")
```

```
Y Antonio dijo: "¿Dónde fue?"
```

CATEGORÍAS

Secuencia	Descripción
\n	Salto de línea
\\	Barra inversa
\'	Apóstrofe (comilla simple)
\"	Comillas
\a	Campana
\t	Tabulado horizontal
\v	Tabulado vertical