INFO1110 Cheatsheet

Abyan Majid

August 30, 2023

Checklist for assessment 1 preparedness

- ☐ Flowcharts
- ☐ Terminal commands
- ☐ Variables, typecasting, inputs
- ☐ Type annotation
- ☐ __name__, __doc____
- ☐ concatenation, multiple print args, format(), f-string(), %*s*
- ☐ comments, docstring
- ☐ capitalize(), upper(), lower(), strip(), replace()
- ☐ len(), count(), type()
- ☐ isalpha(), isnumeric()
- ☐ and, or, not
- ☐ ==, !=, <, >, <=, >=
- ☐ +, -, *, /, **, %, //, ^ (bitwise XOR)
- ☐ if, else, elif, nested ifs
- ☐ string indexing, string slicing
- ☐ function parameters, arguments, return type, body
- ☐ return, pass, None
- ☐ count-controlled while, event-controlled while
- ☐ math lib: e, pi, sqrt(), ceil, floor
- ☐ chr(), ord(), hex(), oct()
- ☐ num:.xf, num:xd
- ☐ debugging: pdb, IDE

---

1 Terminal commands

```
$ python3 <filename>.py   # interpret and run filename.py
$ python3 -m py_compile <filename>  # create a .pyc executable binary
$ python3   # enter python interpreter
$ pwd   # see current directory
$ ls    # see contents of current directory
$ mkdir <dirname>   # create new directory
$ cd <dirname>   # navigate to another directory
$ touch <filename>   # create a new file
$ mv <filename> <destination>   # move a file to another directory
$ cp <filename> <new_filename>   # copy a file
$ vim <optional: filename>  # enter vim, exit with :q
```

2 Variables, type annotation, __name__, __doc__

On variable naming conventions: Be descriptive, lowercase everything, and write spaces as underscores

```
# Python can infer datatypes, hence you are not required to annotate types.
x = 10
```

```
    y: int = 4.25 # with type annotation
    m, n = ("cool", "shorthand")   # initialize multiple vars in a single line

    x = y # "=" is assignment, not equality!

    def some_function():
        """
        documentation of some_function()
        """

    print(x)
    print(m, n)
    print(__name__)   # __name__ fetches the name of current module.
    print(some_function.__doc__)   # __doc__ fetches documentation of some_function()
```

```
    4.25
    cool shorthand
    __main__

        documentation of some_function()
```

3 Strings

3.1 Ways to format strings: concatenation, multiple args, format(), f-string, %s

```
    a, b = ("Hello", "World")

    # String concatenation
    print("Hello" + ", " + "World!")

    # Passing multiple args into print()
    print("Hello,", "World!")

    # format() method
    print("{}, {}!".format(a, b))

    # f-string (using string literals)
    print(f"{a}, {b}!")

    # using %s
    print("%s, %s!"%(a, b))
```

```
    Hello, World!
    Hello, World!
    Hello, World!
    Hello, World!
    Hello, World!
```

3.2 Functions/methods for strings: capitalize(), upper(), lower(), len(), strip(), replace(), count(), isalpha(), isnumeric()

```
print("hello, wORld!")   # prints to the terminal
print("hello, wORld!".capitalize())   # capitalizes the first character
print("hello, wORld!".upper())   # uppercases all characters
print("hello, wORld!".lower())   # lowercases all characters
print(len("hello, wORld!"))   # get length of string
print("hello, wORld!".strip("hello, "))   # removes "hello, " from string
print("hello, wORld!".replace("hello", "Bye"))   # replaces "hello" with "Bye"
print("hhhhheeelloooo!".count("h"))   # counts the number of "h" in the string
print("100".isalpha(), "hello".isalpha())   # checks if all char are in alphabet
print("100".isnumeric(), "hello".isnumeric())   # checks if all char are numeric
```

```
hello, wORld!
Hello, wORld!
HELLO, WORLD!
hello, world!
13
wORld!
Bye, wORld!
5
False True
True False
```

3.3 Indexing strings

```
print("Hello, World!"[0])   # print 1st char
print("Hello, World!"[-1])   # print last char
print("Hello, World!"[:6])   # print chars from the beginning until the 6th (exclusive)
print("Hello, World!"[6:])   # print the 6th chars and onwards
print("Hello, World!"[2:6])   # print 2nd char until the 6th (exclusive)
print("Hello, World!"[::2])   # print alternate chars
```

```
H
!
Hello,
 World!
llo,
Hlo ol!
```

4 Operators

4.1 Arithmetic operators

```
a, b = (10, 4)
print(a + b)   # addition
print(a - b)   # subtraction
print(a * b)   # multiplication
print(a / b)   # division
print(a ** b)   # exponentiation
print(a % b)   # modulus/get remainder
print(a // b)   # floor division/get quotient
print(a ^ b)   # bitwise XOR (prob not assessed)
```

```
14
6
40
2.5
10000
2
2
14
```

4.2 Comparison and boolean operators

Comparison operators in python: ==, !=, <, >, <=, >=
Boolean operators in python: **and, or, not**

```python
a = 1
b = 2
c = b
print(a == b)
print(a != b)
print(a < b)
print(a > b)
print(a <= c)
print(a >= c)
print(not(a == b))
print((b == c) and (a != b))
print((b != c) or (a == b))
```

```
False
True
True
False
True
False
True
True
False
```

5 Inputs, typecasting, encoding: int(), float(), str(), bool(), chr(), ord(), hex(), oct()

```python
foo = input("Some input text: ")   # input
bar = float(input("Enter a float: "))   # typecast str input to float

int_of_bar = int(bar)   # float to int
print(int_of_bar)
print(float(int_of_bar))   # int to float
print(str(int_of_bar))   # int to str
print(str(bar))   # float to io str
print(bool(int_of_bar))   # int to bool, anything > 0 evaluates to True
print(bool(0))   # int 0 to bool
print(chr(97))   # ASCII to char
```

```
print(ord("A"))    # char to ASCII
print(hex(128))    # decimal to hex
print(oct(56))    # decimal to octal
```

```
Some input text: #Hello!
Enter a float: #3.14
3
3.0
3
3.14
True
False
a
65
0x80
0o70
```

6 Conditionals: if, elif, else, nested ifs

```
temperature = 25

if temperature > 30:    # if
    print("It's hot")
elif temperature > 20:    # else if
    print("It's not that hot")
else:    # else
    print("It's a bit chilly")

# nested ifs
if temperature > 20:
    if temperature < 30:
        print("Wear a jacket")
    else:
        print("You may not wear a jacket")
```

```
It's not that hot
Wear a jacket
```

7 while loops: count-controlled, event-controlled, infinite loop

```
# count-controlled while loop
count = 0
while count < 3:    # stops looping when count >= 3
    print(count)
    count += 1

# event-controlled while loop
user_input = ""
while user_input != "exit":    # stops looping when user_input == "exit"
    user_input = input("Enter a value: ")
    print("You entered:", user_input)
```

```
    # infinite loop (discouraged unless absolutely necessary)
    while True:
        # do stuff
```

```
0
1
2
Enter a value: #6
You entered: 6
Enter a value: #8
You entered: 8
Enter a value: #exit
You entered: exit
```

8 Functions

```
def multiply(param1: int, param2):   # type annotation for parameters is omissible.
    return param1 * param2

def return_none(param1, param2) -> int:   # return type annotation
    product = param1 * param2   # no return statement!
    # Therefore, function returns None regardless of return type annotation <int>

arg1 = 5
arg2 = 7
print(multiply(arg1, arg2))
print(return_none(arg1, arg2))
```

```
35
None
```

9 Math library: e, π, $\sqrt{x}$

```
import math   # imports the math library into scope

e = math.e   # euler's constant
pi = math.pi   # pi
root_of_5 = math.sqrt(5)   # square root function

print(e)
print(pi)
print(root_of_5)
print(math.ceil(5.6))
print(math.floor(5.6))
```

```
2.718281828459045
3.141592653589793
2.23606797749979
6
5
```