

Types Branching Functions str, int, float Return value w/ positional param **Basic if** Conditionally execute indent a = "hello!" # string def in_file(name): if cost < 10: count = 3# integer path = "./src/" + name print("impulse buy") return path + ".html" pi = 3.14# float path = in_file("home") **list** ordered collection Boolean operators "and", "or" html = open(path).read() a = ["a", "b", 3]if age > 17 and place == "UK": Keyword parameters # "a." a[0]print("can buy alcohol") # "b" a[1] def greet(name="Jack"): if age < 18 or s == "student": # "3" a[-1]print("Hello", name) print("can get discount") # ["b", 3] a[1:2] greet(name="Jill") tuple same as list, but immutable If-elif-else Variable length arguments a = ("a", "b", 3)if beer == "Darkwing": def do_all(*args, **kwargs): print(kwargs) # kwargs is dict print("IPA") dict collection of keys and values elif beer == "Stonehenge": return sum(args) print("Stout") $do_all(3, 5, b=3)$ a = {"test": 1, "b": "hello"} else: a["test"] # 1 Comment aka "docstring" print("Unknown beer") a["b"] # "hello" del a["test"] # delete "test" def plural(word): a["c"] = 3 # add "c" to dict Pass placeholder that does nothing Return the plural of sets "keys-only dict", with operations if cost > 1.99: an English word. pass # TODO: finish this $a = \{"a", 1, 4, "b"\}$ if word.endswith("s"): $b = \{ "a", "b" \}$ return word + "es" print(a - b) # $\{1, 4\}$ return word + "s" ITERATION print("Many", plural("cat")) list methods Lambda alternative syntax for onea = ["a", "b", 3] a.append(4) # ["a", "b", 3, 4] a.reverse() # [4, 3, "b", "a"] While loop Repeat indented code unliners til condition is no longer true cubed = lambda i: i ** 3 print("5^3 is ", cubed(5)) i = 2while i < 10000: dict methods print("square:", i) a = {"a": 1, "b": 2} a.get("c", 3) # 3 as default a.update({"d": 4}) # add more a.keys() # itemable of heas i = i ** 2More For loop Repeat for each item in itera.keys() # iterable of keys a.values() # ... of values a.items() # ... of both Try / except Handle or ignore errors. able try: big_number = 1 / 0 names = ["John", "Paul", "G"] except Exception as e: for name in names: print("It broke:", e) print("name:", name) for x in range(0, 100): INPUT/OUTPUT With Execute code in a context print("x:", x) Prompt user with open("file.txt") as f: List comprehension Create a new list f.write("test") name = input("Name? ") while looping print("Hi ", name) Unpacking assignment Assign to two or more, good for loops names = ["John", "Paul", "G"] Read from file and convert to str long_names = [x, y = [35, 15] pairs = [(10, 5), (8, 100)] n.lower() for n in names a = open("file.txt").read() if len(n) > 2print("data:", a.decode("utf-8")) for left, right in pairs: # = ["john", "paul"] print(left * right) Write to file creating if none **Interruption** Exit loops prematurely

a = "Some text for o.txt"

open("o.txt", "w+").write(a)

with continue

with break, skip to next iteration