**Алгоритмнің трассировкасы**

Кез келген программалау тілінде жазылған программаны дер кезінде реттеп,**түзетуге (отладка)**болады. Программалау тілінде қателерді түзететін өз құралдары бар. Программалаушылар өз тілінде программада жіберілген қателерді **багтар (ағыл. bug – қате)**, программа кодын түзетуді **дебаггер (ағыл. debugger – түзетуші)** деп атайды. Python тілінде де қатені жөндеп, программаны түзететін, реттейтін өз құралдары бар.

*Программалаушылар программа жазуда 3 түрлі қате жіберуі мүмкін. Олар:*

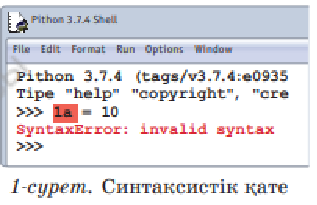
1. *Синтаксистік;*
2. *Программаның орындау уақытына байланысты;*
3. *Алгоритмдік қателер.*

Программаның дұрыс жұмыс істемеу себебін білу үшін, командалардың орындалуының нақты тәртібін білу керек. Оны **программада трассировкалау** деп атайды.

**Трассировкалау** алгоритмі кездесетін қатемен жұмыс істеу ретін білдіреді. Программаны біртіндеп, қадамдап (step-by-step) орындау процесі **трассировка**деп аталады.

**Синтаксистік қате**

1-суреттегі қате айнымалы саннан басталған. Мұндай қателерді программа интерпретация кезінде бірден көрсетеді. Егер қате дер кезінде жөнделмесе, онда программа жұмысын тоқтатады.



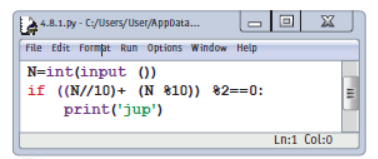
**Программаның орындалу уақытына байланысты қате**

Қателердің тағы бір түрі программаның орындалу уақытының өтіп кетуімен байланысты (run-time errors). Бұл программаны іске қосу немесе тестілеу кезінде пайда болады. Мысалы, бұл қатенің туындау себебі – программа компьютерде жоқ**input.txt** мәтіндік файлын оқуға әрекет жасайды. Бұл файлды таппаған соң, «қате» деп көрсетеді. Программада қате болса, хабарлама беріледі де, тоқтайды.

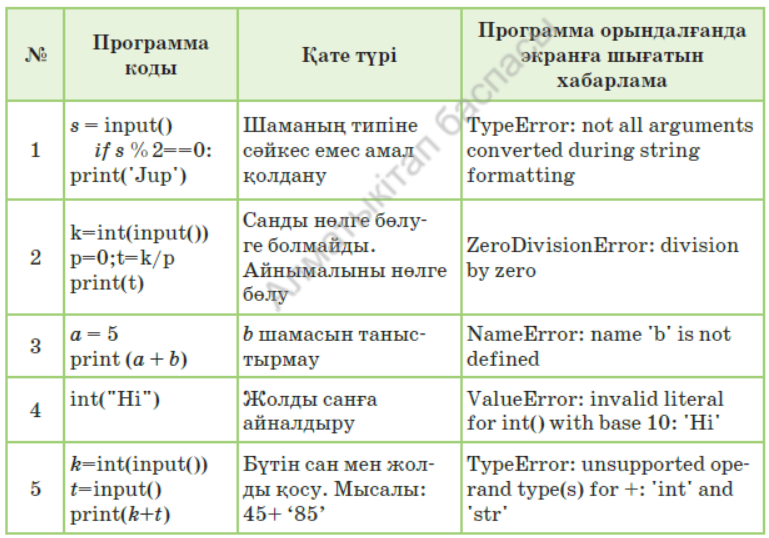
**Алгоритмдік қателер**

**Тестілеу** – программаның жұмыс қабілетін тексеруге арналған, арнайы дайындалған деректерді компьютерге енгізу. Егер программа тестілеу кезінде қарастырылған бір жағдайды дұрыс орындап, басқасын қате орындаса, онда бұл – есептің алгоритмін құрудан кеткен қате. Мысалы, «+» орнына «–» таңбасын немесе**«\*»** белгісін шатастырып қою. Қатенің бұл түрі интерпретация кезінде көрінбейді және программа дұрыс іске қосылады. Бірақ нәтижедегі жауап қате болады. Ол үшін программа кодын қайта қарап, жөндеу керек. Міне, осы процесс **түзету жасау кезі (отладка)**деп аталады. Қарапайым мысалды қарастырайық.

Екітаңбалы сан берілген. Сол санның цифрларының қосындысы жұп немесе тақ екенін анықтайтын жоба жасайық.

Программа коды интерпретацияланып, жұмыс істеп тұр. Бірақ бұл программа коды арқылы тек «жұп» санды ғана анықтауға болады. Программа кодында «тақ» сан болу жағдайы қарастырылмаған. Міне, алгоритмдік қате деген осы **

Программаны қайта қарап, шарт орындалмаған жағдайға **«else: (‘taq’)»** жолын жазсақ, программа коды жөнделіп, «тақ сан» берілгенде де нәтижені дұрыс табады. Сонымен бірге программалау барысында жиі кездесетін бірнеше түрлі қате бар. Соларға тоқталып өтейік:



Программаның қатесін жөндеу үшін, трассировкалау кезінде **quit()** командасын пайдаланған жөн. Цикл денесінің жұмысын бақылау үшін шарт тексеру операторымен бірге қолдану жақсы нәтиже береді.